General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

APPENDIX I

A. Portable Clinical or Laboratory Computation System

   1. General Description

   2. CBL/NOVA Analog Interface: Reference Manual and Assembly Language Programming Guide

   3. CBL/NOVA RT Basic: User Manual and Programming Guide

B. Stereo Display of Vectorcardiograms by Jerome N. Sheridan

NASA CR-
144556

APPENDIX I

(NASA-CR-144556) ENGINEERING STUDIES OF VECTORCARDIOGRAPHS IN BLOOD PRESSURE MEASURING SYSTEMS, APPENDIX 1 Final Report (Trustees of Health and Hospitals of the City) 243 p HC $8.00 CSCL 06B G3/52

N76-12699

Unclas
01917

NOV 1975
RECEIVED
NASA STI FACILITY
INPUT BRANCH

PORTABLE CLINICAL OR LABORATORY COMPUTATION SYSTEM

Excerpted from:

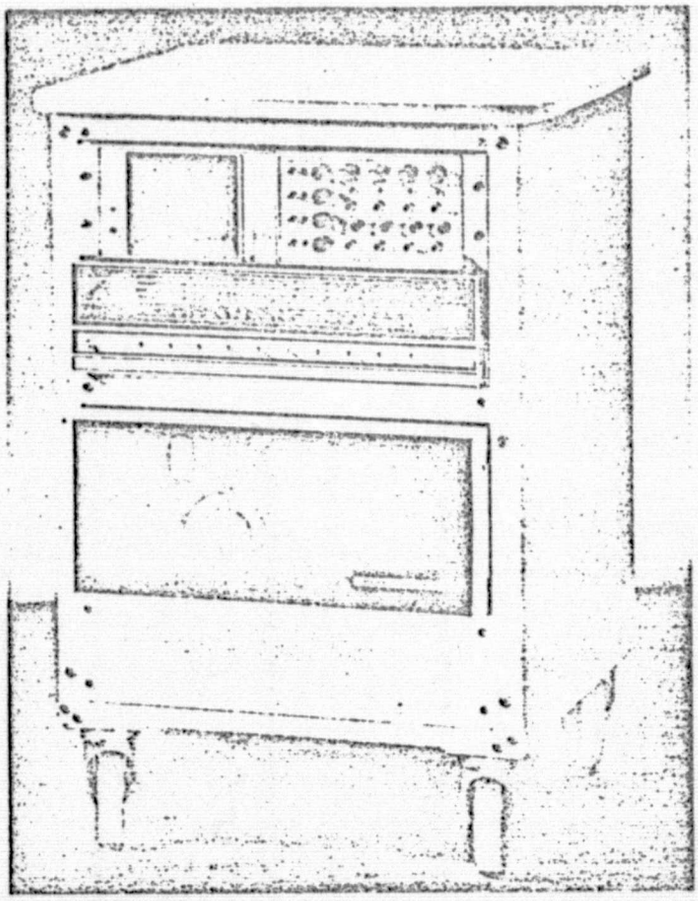
QUARTERLY PROGRESS REPORT NO. 106 July 15, 1972

pp. 172-184

B. PORTABLE CLINICAL OR LABORATORY COMPUTATION SYSTEM

1. Introduction

A small, portable, relatively inexpensive computer system (Fig. XIII-5) has been developed for on-line use in clinical or laboratory situations. The system features an integrated hardware-software package that permits use of all peripherals, such as analog-to-digital converter, oscilloscope, plotter, digital bus, and so forth, with an



ORIGINAL PAGE IS
OF POOR QUALITY

Fig. XIII-5. Portable clinical laboratory computation system.

interpreter constructed around the BASIC programming language.[1] This report gives a brief description of the hardware of the system. Conceptually, the system is similar to the LINC system, which was developed in 1962.[2] This was the earliest computer system designed explicitly for application in a laboratory environment. Since the development of the LINC, advances in integrated circuit technology have made much more compact and powerful systems possible.

The Analog Interface is a system of I/O devices which communicate among the NOVA series of computers (Data General Corporation), the analog world, and the user. combination with the NOVA, this system provides a compact, powerful, easy-to-operate system for processing physiologic data. The interface consists of a 15 × 15 in. main logic board and a 5 1/4 in. half-rack analog chassis (Fig. XIII-6). The main logic board is inside the NOVA main frame and the analog chassis is mounted near the NOVA main frame, sharing rack space with a small X-Y display oscilloscope (Tektronix, Inc.).

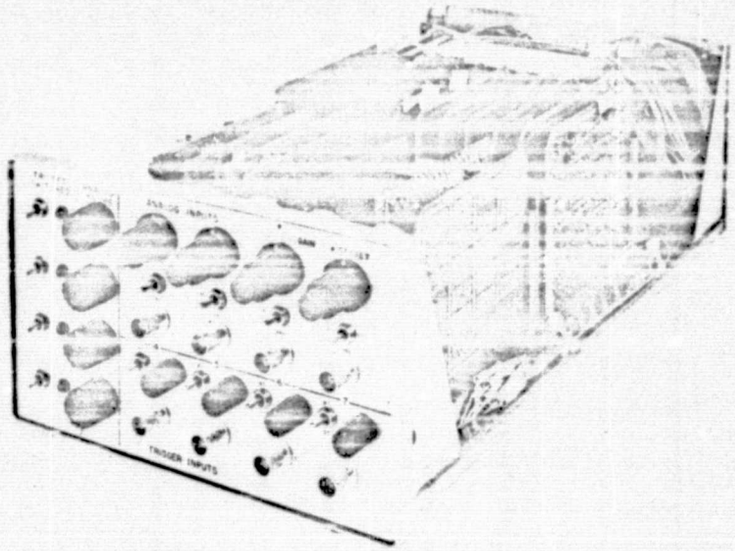


Fig. XIII-6. Analog chassis.

Figure XIII-7 is a block diagram of the system, which is organized around a NOVA line processor and its associated memory. As well as the familiar teletype, special peripherals have been added to enhance the capabilities of the system. The NOVA line includes small general-purpose processors that are compatible from model to model.

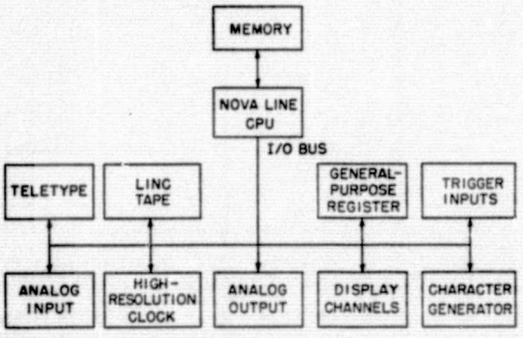


Fig. XIII-7.
Diagram of the clinical laboratory system.

The Central Processing Unit (CPU) is organized around 4 general-purpose registers, two of which may be used as index registers. The NOVA shown in Fig. XIII-5 (NOVA 800) contains 8192 16-bit words of 800-ns memory. At the bottom of the rack is a LINC tape (Computer Operations, Inc.) that provides inexpensive and reliable storage for random access of programs and data. A second system in current use also includes a 256K fixed-head disk for high-speed storage and retrieval. All of the other preipherals shown in Fig. XIII-7 were developed in this laboratory, and will be described in more detail.

ORIGINAL PAGE IS
OF POOR QUALITY

2. Analog Chassis Front Panel

The analog chassis front panel (Fig. XIII-8) provides a convenient point of interaction between the user and the system. It is divided into four sections: analog inputs, analog knobs, trigger inputs, and trigger switches. The analog inputs and analog knobs provide input for 8 of the 16 available analog-input channels. The trigger switches allow the user to set flags in the device corresponding to switches; similarly, the trigger inputs accept analog signals as input and set flags when the input exceeds a set threshold.

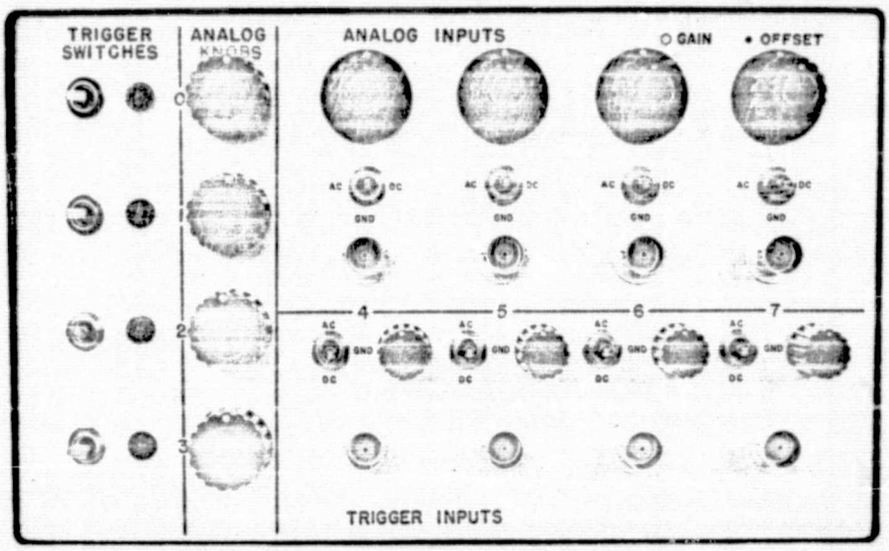


Fig. XIII-8. Analog chassis front panel.

The analog inputs accept analog signals over a range of ±0.2 to ±4 V and allow the user to adjust gain and offset controls to bring the signals into the normal operating range of the analog multiplexer (±2 V). The user may also select ac or dc signal coupling.

The analog knobs provide a convenient method of presenting parameters to a program. A full counterclockwise setting on a knob corresponds to minimum input, while a full clockwise setting corresponds to maximum input.

The trigger inputs accept signals in the range ±10 V. The user may set a threshold voltage; if the signal exceeds this voltage, the corresponding trigger flag will be set. The trigger circuit provides approximately 0.25 V of hysteresis as protection against noise. The user may also select signal coupling at these inputs.

The trigger switches provide a convenient method of setting a program flag. They may be depressed, in which case they spring back to the rest position when released, or they may be pushed upward into the latch position. The switch will remain in the latch

position until the user resets it. A Light-Emitting Diode (LED) lamp is associated with each switch. These lamps are under program control, not electrically connected to the switches. While it is convenient to associate them with the state of a switch, they may indicate any condition the user desires.

All signal inputs at the front panel are fully protected against input voltages of ±30 V. Any input signal outside the normal operating range (~ ±12 V) will activate the protective circuitry. If the protective circuitry is overloaded, the circuit will break down in such a way as to open the signal line, and thus disconnect the overload signal from the signal-conditioning electronics.

3. Special System Peripherals

a. High-Resolution Clock

The High-Resolution Clock (HRC) is a programmable interval clock. The HRC may be programmed to set hardware flags at a periodic rate, or perform interval measurements on an external signal.

Figure XIII-9 is a block diagram of the HRC, which contains three 16-bit registers: the preset, the counting, and the output registers. The counting register is periodically incremented by the HRC control logic's scaled clock. The scaled clock is derived by dividing the selected clock frequency (internal 1 MHz or external clock) by the clock scale factor in the control register. When the counting register overflows (produces a carry-out of the most significant bit), the control logic automatically presets the counting register with the contents of the preset register. Thus, for a given scaled-clock frequency, the period between counter register overflows is determined by the contents of the preset register. By loading the preset register and control register with appropriate values, the user can program the counter register to overflow at precise intervals. The scaled-clock frequency selected by the user determines the basic resolution, while the contents of the preset register determines the interval within that resolution. The major constraints on the user are the number of available scaled-clock frequencies (four) and the fact that the counting register is 16 bits, which limits each period to 65,536 scaled-clock cycles. In normal operation, the output register tracks the counting register to allow a readout of the current clock value.

Each time the counting register overflows, the control logic produces an HRC overflow pulse. This overflow pulse is distributed to the other devices in the analog interface. This pulse allows the activity of the analog interface to be clocked at a precise, periodic rate. It is particularly useful for controlling the sampling interval on the analog-to-digital converter and the conversion intervals for the dual-channel digital-to-analog converters.

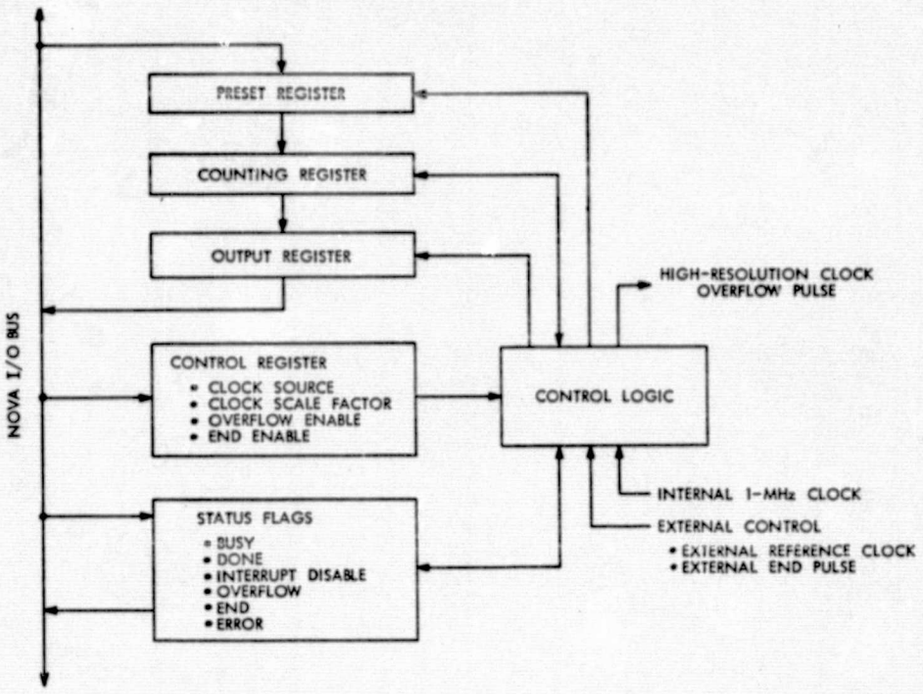


Fig. XIII-9. High-resolution clock. Arrows indicate data and control information flow.

An Overflow Enable feature allows the HRC to request program service when the counting register overflows. By use of this feature, a program can maintain elapsed time. An End Enable feature allows the output register to be frozen upon the occurrence of an external event, and hence provides the capability of interval measurement.

b. Analog Input

The Analog Input (AIN) provides 16 analog input channels. Any channel may be selected under program control to yield a 10-bit sample. Figure XIII-10 is a block diagram of the system. The front end of the device has channel conditioners to drive the 16-channel multiplexer. The output of the multiplexer is sampled by the Sample-and-Hold which drives the 10-bit analog-to-digital converter. The output of the analog-to-digital converter may be read in by the user. In addition to the analog section, a digital section is included containing the channel select register, the control register, the status flags, and the control logic.

The contents of the channel select register determines which of the input channels is selected by the multiplexer for the Sample-and-Hold. If the user selects the scan mode, the content of the channel select register is automatically incremented after each conversion, so that the next sequential channel is selected.

The control register determines the mode of operation of the AIN. It allows the user

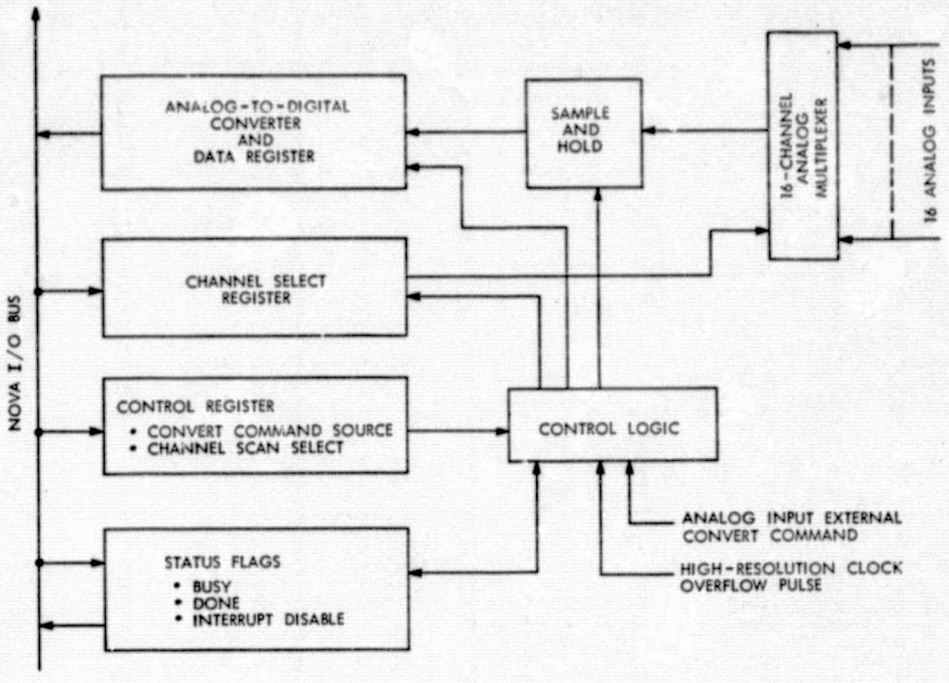


Fig. XIII-10. Analog input.

to specify the source of the command in order to initiate a new conversion cycle, as well as to select a scanning or nonscanning mode. The user may select, as the source of the conversion command, the program, the HRC overflow pulse, or an external control line.

A conversion cycle comprises the following events: First, the request for a conversion cycle is noted by the hardware. When the multiplexer has selected the requested channel (as specified by the channel select register) the conversion is initiated by sampling the output of the multiplexer. After the Sample-and-Hold has acquired the signal and "frozen" it, the analog-to-digital converter begins the actual conversion to a digital representation. Since the Sample-and-Hold maintains the signal value, the multiplexer may be released to scan to a new channel, if this mode is selected. When the converter finishes, the AIN will indicate that a valid conversion value is available to the user.

c. Dual Channel Analog Output

The Dual Channel Analog Output (AOUT) comprises two 10-bit digital-to-analog converters and the associated control logic. The AOUT (Fig. XIII-11) allows the user to generate two channels of analog output in the ±2 V range. Output conversions may be initiated under direct program control from the HRC overflow pulse, or from an external control line.

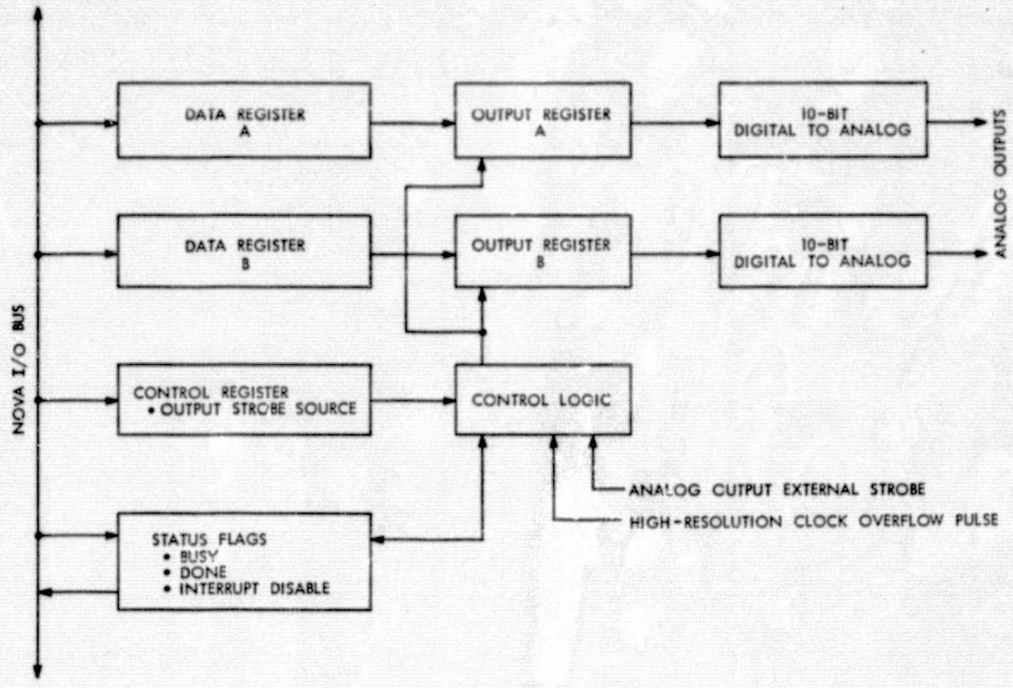


Fig. XIII-11. Analog output.

The data path for each channel goes from the program (by way of the NOVA bus) to the channel data register, from the data register to the output register, and finally to the 10-bit digital-to-analog converter. The user may load the data register directly, while the AOUT control logic loads the output register. While the data registers are loaded independently, the output registers are loaded by the same signal, so that any changes in output are synchronous. The control register selects the source of the signal to initiate the load signal to the output register. Status flags are provided to indicate the current status of the device and to request program service.

The use of two buffers in each channel (double buffering) allows the user to set up the desired output value in advance of the actual time it will be required, without changing the previous output value.

d. Display Hardware

The Display (DISP) provides the capability of generating displays. The DISP has two 10-bit digital-to-analog converters, and provides an addressable field of 1024 × 1024 points. As many as four X-Y display devices may be attached to the DISP output bus. The same X-Y deflection signals are applied to each device, but the devices are individually addressable for beam intensification.

Figure XIII-12 is a block diagram of the DISP. The unit is similar to the AOUT,

A display output device may also indicate that it is not ready to accept changes in the values of X-Y by activating the Display Wait line. The DISP control logic will not change the contents of the output registers while this signal is active, and thus prevents change in the X-Y analog signals. If the control logic receives a request to load the output registers while Display Wait is active, it will honor the request as soon as Display Wait becomes nonactive. Typical applications of the Wait feature are raising and lowering the pen on a hardcopy plotter or preventing writing on a storage oscilloscope while the screen is being erased.

e. Character Generator

The hardware Character Generator (CHAR) provides the user with a system for the generation of characters on the display output devices. The CHAR can represent 64 ASCII characters by a 5 × 7 dot matrix. The hardware in use at the present time is an improved version of a character generator described in an earlier report.[3]

Figure XIII-13 is a block diagram of the character generator. The user may access the character register, size register, and status flags. The character register specifies the character to be drawn, while the size register specifies the physical size of the character on the display device. The status flags are shared with the DISP logic,

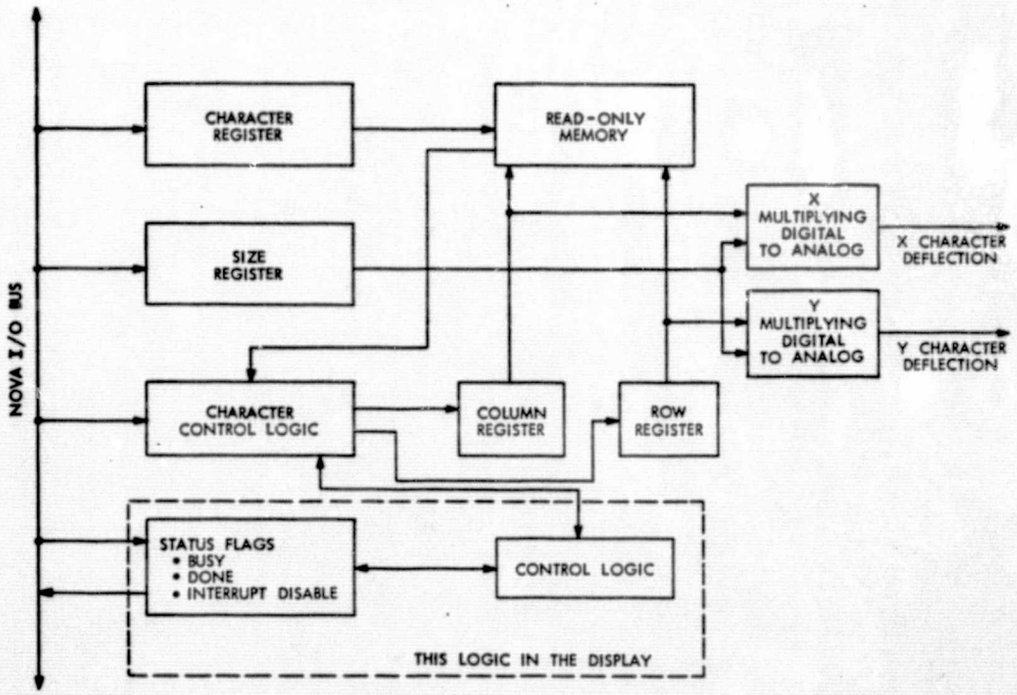


Fig. XIII-13. Character generator.

PRECEDING PAGE BLANK NOT FILMED

so that the user need not remember which device was the last to carry out display activity, nor need he test the status of both DISP and CHAR. All the user need do is verify that the device he wishes to use is free.

The column and row registers (which are not accessible to the user) specify the current column and row address of the dot that is to be plotted. Together with the character code, these registers provide address information for the Read Only Memory. The output of the Read Only Memory for a given character, column and row determines whether or not a dot will be drawn on the display. The column and row registers specify the relative deflections of the X and Y outputs, and thus scan the character field. In Fig. XIII-12 the deflection signals from the CHAR are summed with the current X and Y coordinates to determine the actual position of the character on the display field.

f. General-Purpose Digital Register

The General-Purpose Digital Register (GPR) provides 16 lines of digital input and 16 lines of digital output for the user. The GPR allows the user to interface to digitally programmed hardware that is not used often enough to justify construction of a special-purpose interface to the NOVA.

Figure XIII-14 is a block diagram of the GPR. A 16-bit digital output register is

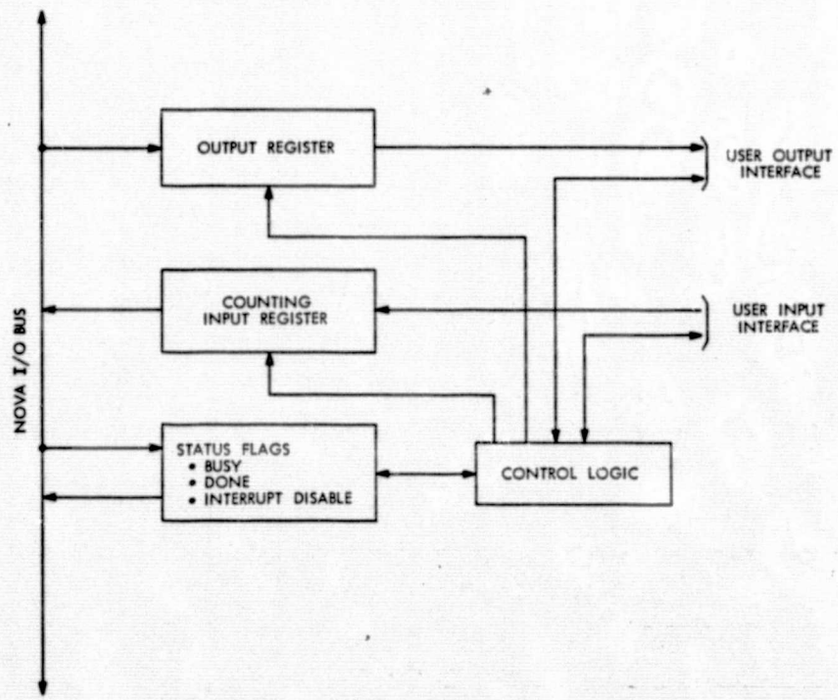


Fig. XIII-14. General-purpose register.

provided. The programmer may load this register and the contents remains fixed until changed by the user. The 16-bit counting input register will accept 16 parallel digital inputs from a user interface. It may also be incremented by the user interface for use as an event counter.

The control logic allows control of the status flags by the user interface, so that the interface may request program service as would any special-purpose interface.

g. Trigger Input

The Trigger Input (TRG) provides a facility for interaction between the user and program. Four front-panel switches are provided, and when the user trips any of these switches a unique hardware flag is set which may be interrogated by the program. Analog trigger inputs are also provided. Each of these inputs allows an analog signal to set a unique trigger flag, which can be interrogated in a manner similar to the switch flags. Thus a wide variety of signals may be used to flag events for the program.

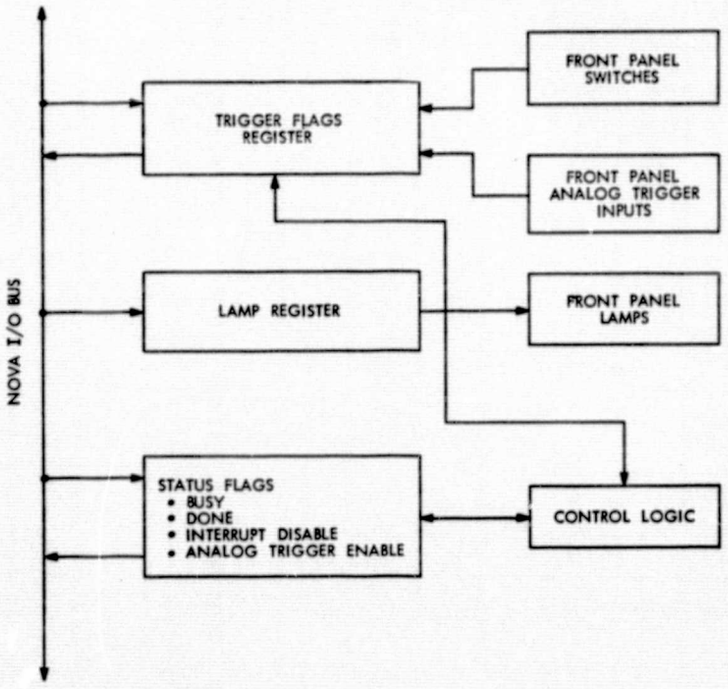


Fig. XIII-15. Trigger input.

Figure XIII-15 is a block diagram of the TRG system. The TRG is represented on the front panel by 4 switches, 4 analog trigger inputs, and 4 LED lamps (see Fig. XIII-8). The various inputs set flag bits in the trigger flags register. The user may load the lamp register, and thus activate a LED lamp on the front panel. The control logic responds

to changes in the status of the trigger flags register (and thus indirectly in the various inputs) and sets the status flags to request program service.

Each of the analog trigger inputs accepts analog inputs in the ±10 V range and compares the input with a reference value determined by the threshold knob (above and to the right of the input, see Fig. XIII-8). When the input signal exceeds the reference, an event is said to have occurred on that input. Thus a trigger flag is set on the occurrence of an event, and will remain set even after completion of the event. If another event occurs while the corresponding trigger flag is set, the flag will remain set.

The implementation of this system would not have been possible without the guidance and encouragement of Professor S. K. Burns, who also designed the hardware character generator. The General-Purpose Register was designed by Barry Gaiman, who also provided many useful comments about programming the system during the development stage.

J. B. Walters, Jr.

References

1. B. Gaiman, "A Real-time Digital Processing System for Electrophysiologic Waveforms," S. B. and S. M. Thesis, Department of Electrical Engineering, M. I. T., September 1970.
2. W. A. Clark and C. E. Molnar, "A Description of the LINC," in R. W. Stacy and B. D. Waxman (Eds.), Computers in Biomedical Research, Vol. II (Academic Press, Inc., New York, 1965).
3. S. K. Burns, "Use of a Hardware Character Generator with Small Computers," Quarterly Progress Report No. 98, Research Laboratory of Electronics, M. I. T., July 15, 1970, pp. 206-214.

C. TIME-DOMAIN FILTERS

1. Introduction

The trend toward computer monitoring and analysis of the electrocardiogram (ECG) has led to the development of programmed digital filters to eliminate noise prior to detection of the QRS complex and analysis of morphology. Multipole frequency-domain filters are adequate for both tasks but, in general, require large amounts of processor time.

Improved detection for the QRS complex in question can be achieved by using a matched filter, but implementation of such a filter by correlation with a template of a QRS complex requires many times more processor time than is required for frequency-domain filters.

An improvement in the speed of such a filter can be accomplished by a simplification of the template. The QRS can be well approximated by a triangular-shaped wave of

CBL/NOVA ANALOG INTERFACE:
Reference Manual and
Assembly Language Programming Guide

April 1973

J. B. Walters
B. Gaiman

Bioengineering Group
Communications Biophysics Group
Research Laboratory of Electronics
Massachusetts Institute of Technology

Copyright 1973
All Rights Reserved

PRECEDING PAGE BLANK NOT FILMED

CBL/NOVA ANALOG INTERFACE:
Reference Manual and
Assembly Language Programming Guide

April 1973

J. B. Walters
B. Gaiman

Bioengineering Group
Communications Biophysics Group
Research Laboratory of Electronics
Massachusetts Institute of Technology

Copyright 1973
All Rights Reserved

CONTENTS

PREFACE.....7

1. INTRODUCTION TO THE ANALOG INTERFACE
   - 1.0 General Information.....1-1
   - 1.1 Front Panel.....1-7
   - 1.2 How to Use This Manual.....1-8

2. THE HIGH RESOLUTION CLOCK
   - 2.0 Introduction.....2-1
   - 2.1 Preset, Counting, and Output Registers.....2-4
   - 2.2 Control Register.....2-6
   - 2.3 Device Status Flags.....2-9
   - 2.4 Programming the HRC.....2-13

3. THE ANALOG INPUT
   - 3.0 General Information.....3-1
   - 3.1 Input Specifications.....3-4
   - 3.2 Data Register.....3-6
   - 3.3 Channel Select Register.....3-6
   - 3.4 Control Register.....3-7
   - 3.5 Status Flags.....3-9
   - 3.6 Analog Input Timing.....3-11
   - 3.7 Programming the AIN.....3-12

4. THE DUAL CHANNEL ANALOG OUTPUT
   - 4.0 General Information.....4-1
   - 4.1 Analog Outputs.....4-3
   - 4.2 Data Registers.....4-3
   - 4.3 Control Register.....4-4
   - 4.4 Status Flags.....4-6
   - 4.5 Programming the AOUT.....4-9

5. THE DISPLAY ANALOG OUTPUT
   - 5.0 General Information.....5-1
   - 5.1 X-Y Outputs.....5-5
   - 5.2 Data Registers.....5-6
   - 5.3 Display Field.....5-6
   - 5.4 Control Register.....5-8
   - 5.5 Status Flags.....5-10
   - 5.6 Standard Display Devices.....5-13
   - 5.7 Programming the DISP.....5-14

6. THE CHARACTER GENERATOR
   - 6.0 General Information.....6-1
   - 6.1 Character Register.....6-4
   - 6.2 Size Register.....6-4
   - 6.3 Status Flags.....6-7
   - 6.4 Programming the CHAR.....6-9

CONTENTS

(Continued)

7. THE GENERAL PURPOSE REGISTER
   - 7.0 General Information.....7-1
   - 7.1 Output Register.....7-1
   - 7.2 Input Register.....7-3
   - 7.3 Status Flags.....7-4
   - 7.4 Programming the GPR.....7-6

8. THE TRIGGER INPUTS
   - 8.0 General Information.....8-1
   - 8.1 Analog Signal Input Specifications.....8-4
   - 8.2 Trigger Flag Registers.....8-5
   - 8.3 Lamp Register.....8-6
   - 8.4 Status Flags.....8-7
   - 8.5 Programming the TRG.....8-5

APPENDICES

A. Special Assembly Language Definitions.....A-2

B. General Physical and Electrical Specifications.....B-2

C. Interface Signal Specifications and Connector Allocation

C.1 Driving and Receiving External Lines.....C-2
C.2 HRC Signal Specifications.....C-5
C.3 AIN Signal Specifications.....C-6
C.4 AOUT Signal Specifications.....C-8
C.5 DISP Signal Specifications.....C-9
C.6 GPR Signal Specifications.....C-11
C.7 TRG Signal Specifications.....C-13
C.8 GPR Input Connector.....C-14
C.9 GPR Output Connector.....C-15
C.10 TRG Connector.....C-16
C.11 AIN Connector.....C-17
C.12 AOUT Connector.....C-18
C.13 DISP Connectors.....C-19
C.14 Connector Hardware and Tools.....C-20

LIST OF FIGURES

1.1 Portable Clinical Laboratory Computation System.....1-2
1.2 Summary of Analog Interface Features.....1-3
1.3 Analog Chassis.....1-4
1.4 Analog Chassis Front Panel.....1-6

2.1 HRC Block Diagram.....2-2
2.2 HRC Control Register Bit Assignments.....2-7
2.3 HRC Timing Range For Internal 1 M Hz. Clock.....2-8
2.4 HRC Busy-Done State Transitions.....2-10

3.1 AIN Block Diagram.....3-2
3.2 AIN Control Register Bit Assignments.....3-8
3.3 AIN Busy-Done State Transitions.....3-10

4.1 AOUT Block Diagram.....4-2
4.2 AOUT Control Register Bi+ Assignments.....4-5
4.3 AOUT Busy-Done State Transitions.....4-7

5.1 DISP Block Diagram.....5-2
5.2 DISP Coordinate System.....5-7
5.3 DISP Control Register Bit Assignments.....5-9
5.4 DISP Busy-Done State Transitions.....5-11

6.1 CHAR Block Diagram.....6-2
6.2 CHAR Dot Matrix.....6-3
6.3 CHAR Character Set.....6-5
6.4 CHAR Character Size.....6-6
6.5 CHAR Busy-Done State Transitions.....6-8

7.1 GPR Block Diagram.....7-2
7.2 GPR Busy-Done State Transitions.....7-5

8.1 TRG Block Diagram.....8-2
8.2 TRG Busy-Done State Transitions.....8-8

LIST OF PROGRAMMING EXAMPLES

2.1 Routine to Setup HRC.....2-14
2.2 A Sideral Core Clock.....2-16

3.1 Sampling an Analog Signal.....3-15

4.1 Generating an Analog Output.....4-10

5.1 Plot a Function in Y.....5-16

6.1 Displaying a Character.....6-11

7.1 GPR Keyboard Interface.....7-9

8.1 Encoding the Trigger/Latch Inputs.....8-11

PREFACE

This manual describes the CBL/NOVA Analog Interface. General operating instructions and the requisite information for applications programming in NOVA assembler language are provided.

The Analog Interface is a system of I/O devices which provide an interface between the NOVA line of computers, the analog world, and the user. It is designed to facilitate processing of analog data in the laboratory and clinical environment. Included are facilities for sampling and creation of analog data. Waveforms and alphanumeric data can be conveniently presented on displays or on an x-y plotter. A generalized digital signal interface is provided, allowing simple interfacing to digitally programmable hardware.

To take full advantage of the information presented in this manual, the reader should be experienced in NOVA assembler language programming. The reader who is interested in programming the Analog Interface from a higher level language will find Section 1, Introduction to the Analog Interface, to be a sufficient introduction to the system (in addition to the reference manual for the language).

SECTION 1
INTRODUCTION TO THE ANALOG INTERFACE

1.0 General Information

The Analog Interface is a system of I/O devices which provides an interface between the NOVA series of computers, the analog world, and the user. In combination with the NOVA, this system provides a compact, powerful, yet easily used system for processing low frequency data in real time (See Figure 1.1). The Analog Interface consists of seven devices:

1) An Analog to Digital Converter,
2) A High Resolution Clock,
3) An Analog Output,
4) A Dual Channel Analog Output with Display Control Logic,
5) A 5x7 Dot Matrix Hardware Character Generator,
6) A 16 bit digital I/O General Purpose Register, and
7) A Switch and Analog Signal Trigger Register.

The characteristics of these devices are summarized in Figure 1.2.

Physically, the interface consists of a 15x15 inch main logic board and a 5-1/4 inch half-rack analog chassis (Figure 1.3). The main logic board is contained inside the NOVA main frame, while the analog cabinet is mounted near the NOVA sharing rack space with a small X-Y display scope (Figure 1.1).

PRECEDING PAGE BLANK NOT FILMED

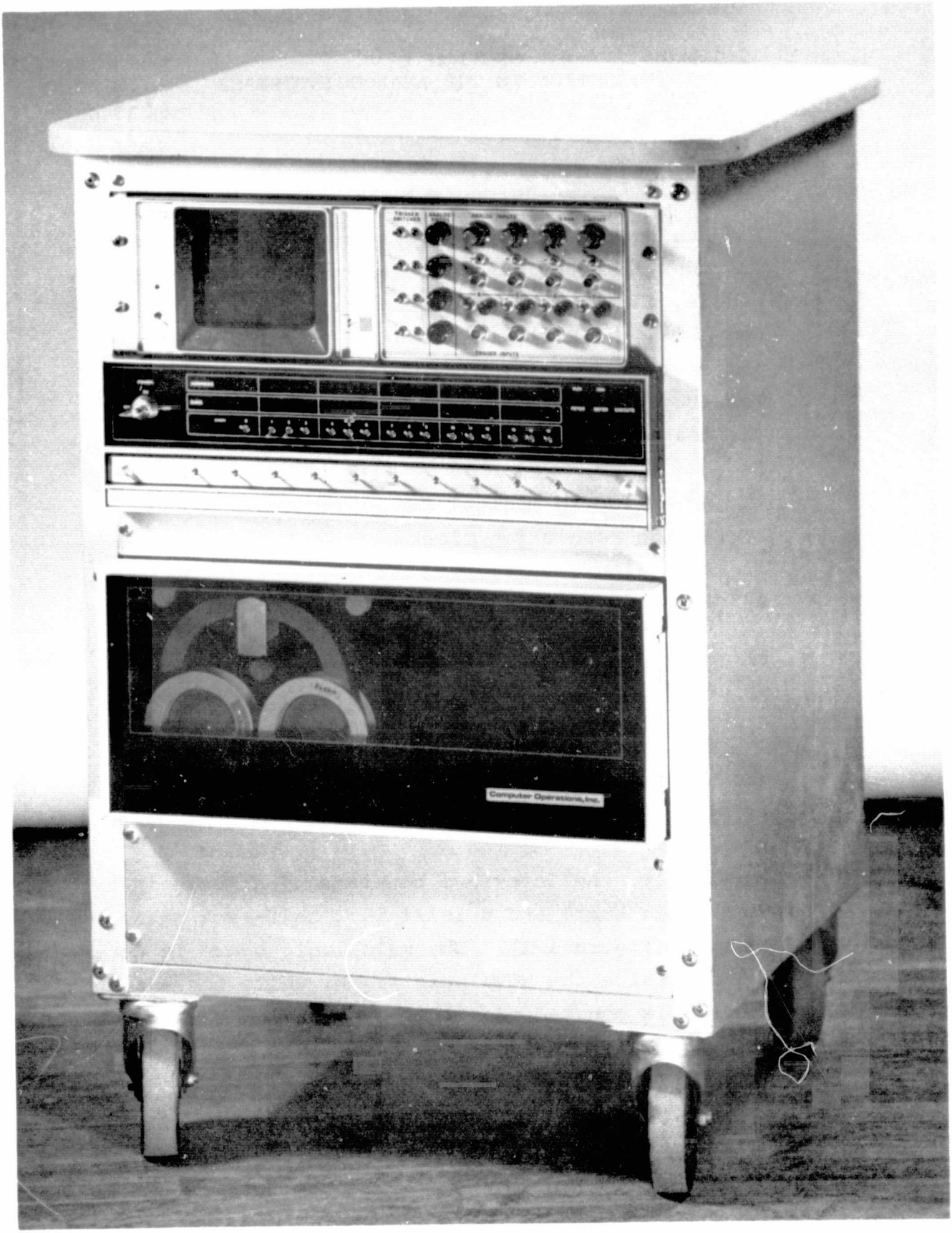


Figure 1.1 - Portable Clinical Laboratory Computation System

1) Analog to Digital Converter (AIN)

-Ten bit resolution
-Sixteen channels of analog input
-4 channels connected to front panel knobs
-4 channels with preamplifiers providing adjustable gain and offset
-Convert command source under program control.

2) High Resolution Clock (HRC)

-16 bit programmable interval clock
-Internal 1 M Hz crystal reference clock
-Resolution of 1 microsec to 1 millisec available from internal clock under program control.
-External reference clock input is available.

3) Analog Output (AOUT)

-Dual 10 bit D/A converters
-Double buffered to prevent output jitter
-Convert command source under program control.

4) X-Y Analog Output (DISP)

-Dual 10 bit D/A converters
-Convert/intensify command source under program control
-4 display channels
-Can extend plotting cycle to accomodate slow peripherals
-Double buffered

5) Hardware Character Generator (CHAR)

-5x7 dot matrix character generator
-Generates 64 ASCII characters
-Program selectable sizes

6) General Purpose Register (GPR)

-16 bit input and output buffers
-Counting input buffer
-Control pulses

7) Trigger Input (TRG)

-4 analog signal inputs with adjustable threshold levels
-High input impedance
-4 front panel toggle switches
-4 LED lamp outputs for feedback to user

Figure 1.2 - Summary of Analog Interface Features

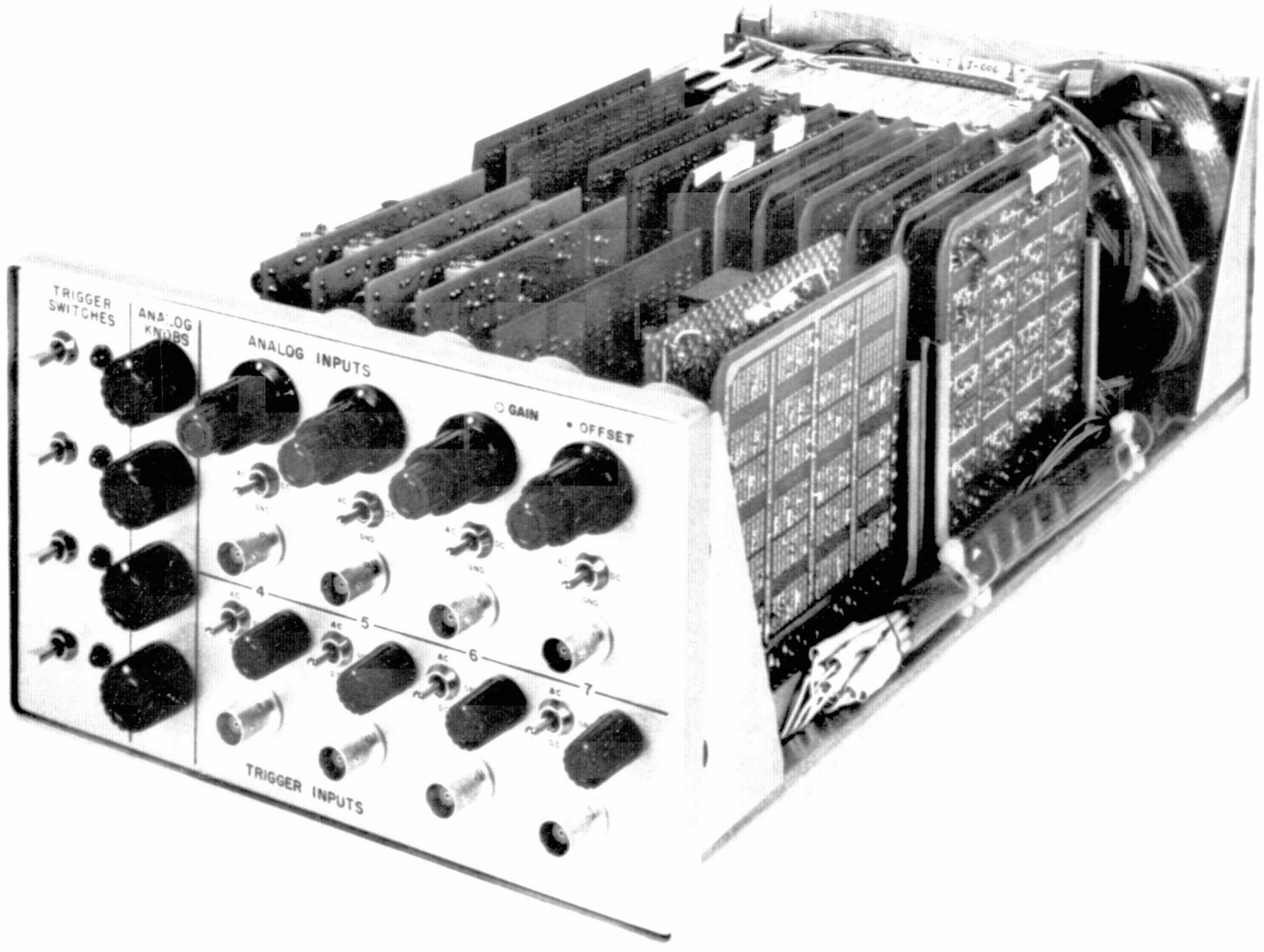


Figure 1.3 - Analog Chassis

1.1 Front Panel

The analog chassis front panel (Fig. 1.4) provides a convenient point of interaction between the user and the system. It is divided into four sections: analog inputs, analog knobs, trigger inputs, and trigger switches. The analog inputs and analog knobs provide input for 8 of the 16 available analog-input channels. The trigger switches allow the user to set corresponding flags in the device. Similarly, the trigger inputs accept analog signals as input and set flags when the input first exceeds a set threshold.

The analog inputs accept analog signals over a range of $\pm$ 0.2 to $\pm$ 4V and allow the user to adjust gain and offset controls to bring the signals into the normal operating range of the analog multiplexer ($\pm$2V). The user may also select direct or capacitive coupling.

The analog knobs provide a convenient method of presenting parameters to a program. A full counter-clockwise setting on a knob corresponds to minimum input, while a full clockwise setting corresponds to maximum input.

The trigger inputs accept signals in the range $\pm$ 10V. The user may set a threshold voltage; if the signal exceeds this voltage, the corresponding trigger flag will be set. The trigger circuit provides approximately 0.25V of hysteresis as protection against noise. The user may also select signal coupling at these inputs.

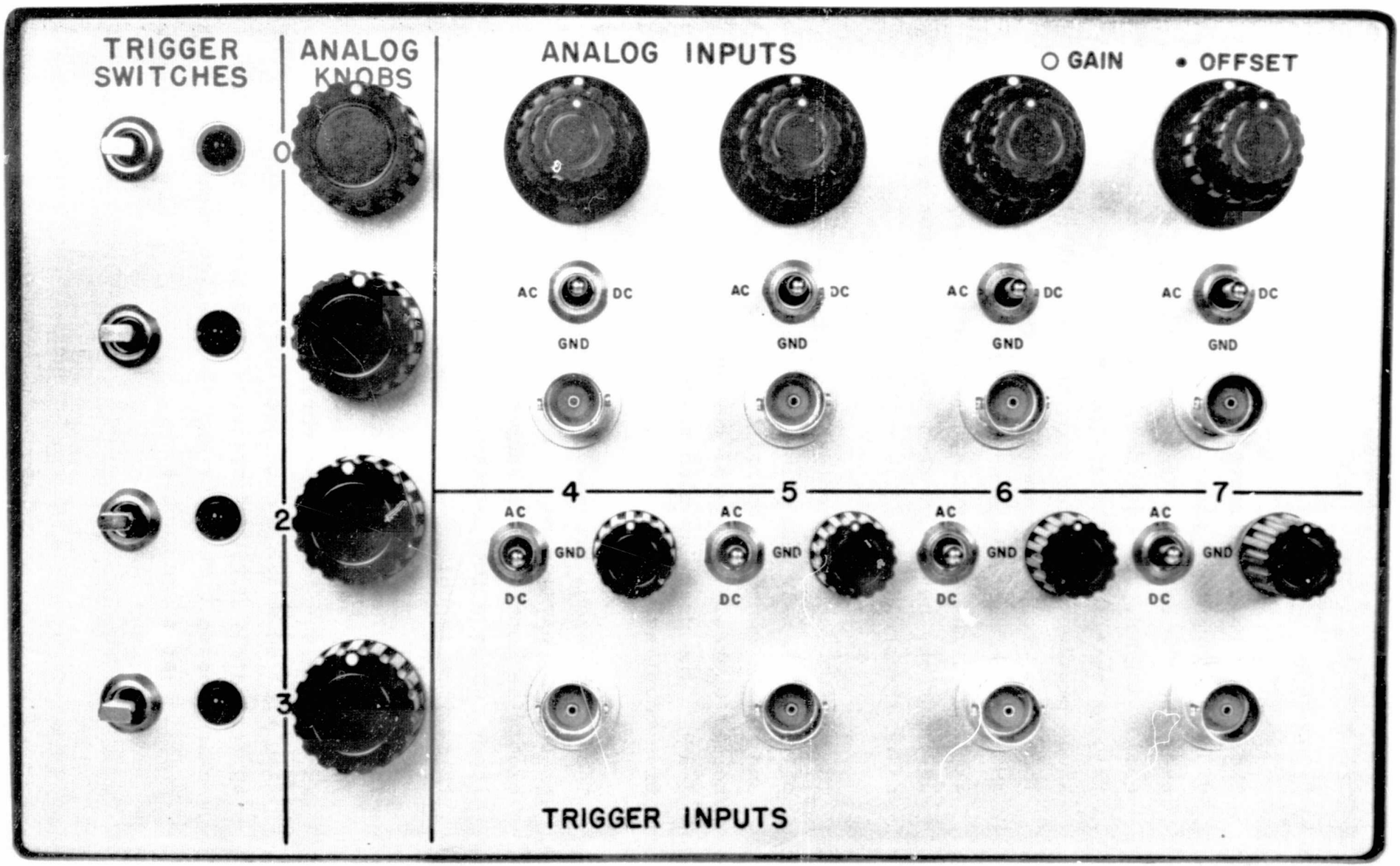


Figure 1.4 - Analog Chassis Front Panel

The trigger switches provide a convenient method of setting a program flag. They may be depressed, in which case they spring back to the rest position when released, or they may be pushed upward into the latch position. The switch will remain in the latch position until the user resets it. A Light-Emitting Diode (LED) lamp is associated with each switch. These lamps are under program control, not electrically connected to the switches. While it is convenient to associate them with the state of a switch, they may indicate any condition the user desires.

All signal inputs at the front panel are fully protected against input voltages of $\pm$ 30V. Any input signal outside the normal operating range ($\sim \pm$12V) will activate the protective circuitry. If the protective circuitry is overloaded, the circuit will break down in such a way as to open the signal line, and thus disconnect the overload signal from the signal-conditioning electronics.

1.2 How to Use This Manual

To take full advantage of the information presented in this manual, the reader should be familiar with the material presented in How to Use the NOVA Computers, particularly Section 2.3 (dealing with the I/O instruction set). The user should also be generally familiar with the contents of the Data General Assembler Manual. If the reader is interested in programming the devices to run in interrupt mode, he should be familiar with Section 2.4 Program Interrupt, of How to Use the NOVA Computers.

The notation for instructions used in explanations of the function of the Analog Interface is the same as used in the Data General Assembler manual. All numbers that appear in the body of the text are decimal unless explicitly indicated otherwise. Numbers that appear in the body of an instruction are octal. Any other radix appearing in an instruction will be marked as such by an accepted assembler convention. In tables and text where it is apparent that groups of control bits are being referred to, the numbers are in binary. The assembler convention of nBm is used. In this notation, the number n will be displaced so that its least significant bit is in position m of the 16 bit word. The results are equivalent to evaluating the expression:

$$n * 2^{(15-m)}$$

Thus, 1BØ produces a word with bit Ø set to 1; 1B15 is the same as 1.

The notation 'C(X)' is used to indicate the contents of X. Thus (C(AC1)) is the contents of accumulator number 1.

The notation "=X" is used to indicate a constant within the address range of the instruction it occurs in.

Thus,

```
LDA 0, = 4
```

indicates load AC with the constant 4.
The instruction,

```
JSR @ = .DAOUT
```

indicates a JSR instruction indirect through a word containing the address of the routine .DAOUT.

A number of mnemonics for various constants useful in programming the Analog Interface are referred to in the text. These mnemonics are defined in "CLDEFS", as ASCII source tape or file if DOS is being used. A listing of this file forms Appendix A. In general, mnemonics that refer to single bits are defined only for the "1" mode. The mode of operation selected by the corresponding bit being "Ø" is considered the default mode. In all cases, the default mode is detailed in CBLDEFS as well as the main text.

SECTION 2

THE HIGH RESOLUTION CLOCK

2.0 General Information

The High Resolution Clock (mnemonic HRC) is a programmable interval clock. The HRC may be programmed to set hardware flags at a periodic rate or perform interval measurements on an external signal.

Figure 2.1 provides a block diagram of the organization of the HRC. The HRC contains three 16 bit registers: the Preset Register, the Counting Register, and the Output Register. The Counting Register is periodically incremented by the HRC Control Logic's scaled clock. The scaled clock is derived by dividing the selected clock frequency (internal 1MHz or External Clock) by the clock scale factor in the Control Register. When the Counting Register overflows (counts from 177777_8 to 000000_8), the Control Logic automatically presets the Counting Register with the contents of the Preset Register. Thus, for a given scaled clock frequency, the period between Counter Register overflows is determined by the contents of the Preset Register. By loading the Preset Register and Control Register with the appropriate values, the user can program the Counter Register to overflow at precise intervals. The scaled clock frequency the user selects determines the basic resolution, while the contents of the Preset Register determine the interval. The major constrains on the user are the available scaled clock frequencies (4) and the length of the Counting Register (16 bits) which limits each period to 65,536

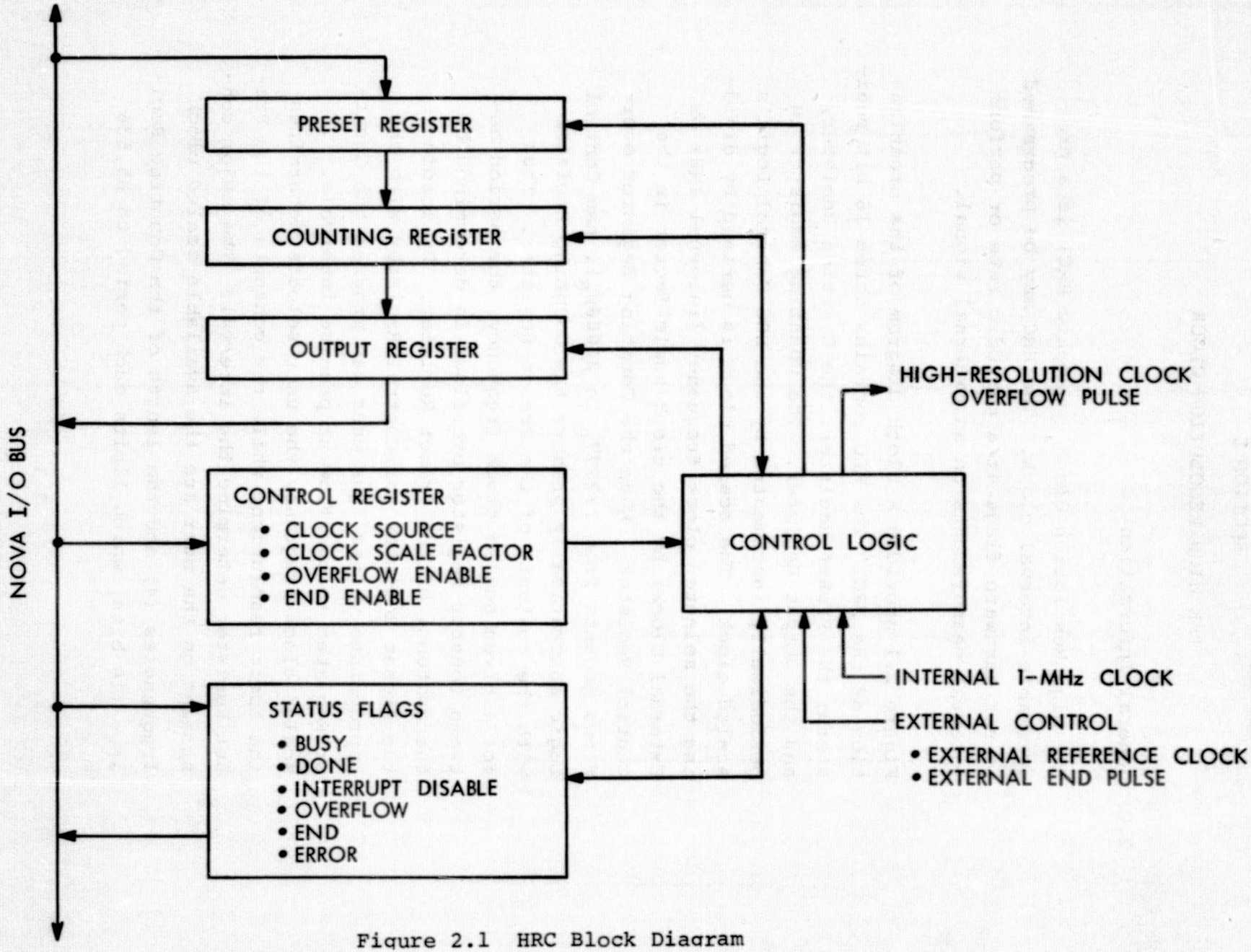


Figure 2.1 HRC Block Diagram

scaled clock cycles. In normal operation, the Output Register tracks the Counting Register to allow a readout of the current clock value.

Each time the Counting Register overflows, the Control Logic produces an HRC Overflow Pulse. This Overflow Pulse is distributed to the other devices in the Analog Interface, allowing the activity of the Analog Interface to be clocked at a precise, periodic rate. It is particularly useful for controlling sampling interval on the analog-to-digital converter and conversion intervals for the dual channel digital-to-analog converters.

The Overflow Enable feature of the HRC allows the unit to perform as a programmable interval clock for maintaining elapsed time. The End Enable feature allows the Output Register to retain the value of the Counting Register upon the occurrence of an external event. This feature allows the user to measure intervals between events with a high degree of accuracy.

2.1 Preset, Counting, and Output Registers

The Preset, Counting, and Output Registers (see Figure 2.1) may be simultaneously set at any time with a DOA s, HRC instruction. The registers are 16 bits wide. The Counting Register is incremented by the Control Logic and is not directly readable by the user. The Output Register, which may be read with a DIA d, HRC instruction, automatically tracks the Counter Register in normal operation. Thus, if the HRC is idle (i.e., after an IORST), after the instruction sequence:

```
DOA Ø, HRC          ;LOAD HRC PRSET REGISTER
DIA 1, HRC          ;READ CONTENTS OF THE HRC OUTPUT
                     REGISTER
```

ACØ and AC1 will contain the same number. If the clock had been running, it could have incremented the Counter Register between the two instructions resulting in different values for ACØ and AC1.

The Counting Register is incremented by the Control Logic at a rate determined by the contents of the Control Register and the selected reference clock. When the Counter Register overflows (counts from 177777_8 to $ØØØØØØ_8$) the Counter Register is preset with the contents of the Preset Register. In addition, an HRC Overflow Pulse is generated. This pulse may be used by other devices in the Analog Interface. Since the Counting Register is reloaded when an overflow occurs, the Preset Register determines the interval in scaled clock cycles between Counting Register overflows. Suppose one wanted the HRC to overflow every 1000 scaled clock cycles.

The following code would set the Preset Register for this mode of operation.

```
NIOC HRC                    ;CLEAR THE HRC

LDA Ø,=1ØØØ                 ;GET DECIMAL 1ØØØ CONSTANT
NEG Ø,Ø                     ;REALLY WANT NEGATIVE OF 1ØØØ

DOA Ø,HRC                   ;SET THE HRC PRESET REGISTER
```

The code starts by clearing the status flags in the HRC, as a matter of good programming practice. Then the decimal 1000 constant is laoded into the Preset Register. Note that we load the register with the negative of the number of clock cycles we wish.

In the Counter Register bit zero is treated as a magnitude bit, not a sign bit. Thus, the range of the HRC registers is considered to be from Ø to 65,535 (177777 octal). If the Counter is loaded with a zero, 65,536 scaled clock cycles are required to produce an overflow condition. If the Counting Register is loaded with 65,535; 1 scaled clock cycle is required to produce an overflow condition.

2.2 Control Register

The contents of the Control Register determines the mode of operation of the HRC. The Control Register must be loaded by the program before attempting to start the HRC. It is not modified by an IORST or C pulse to the HRC.

Figure 2.2 summarizes the options available to the user. Overflow Enable, bit 11 (mnemonic HRCOVFL), allows the user to specify whether or not DONE will be set on the occurrence of an HRC Overflow pulse. End Enable, bit 12 (mnemonic HRCEND), performs the same function for the END FLAG. Note that if both bits 11 and 12 are zero, the HRC will never set DONE. It will however generate HRC Overflow pulses (if "BUSY") which other devices (such as the Analog Input) may use to initiate action. The Clock Source Bit, 13 (Mnemonic HRCEXT), selects the internal 1 MHz. reference clock if it is zero, otherwise the external clock input is selected. The Clock Scale Factor, bits 14 and 15, select the scale factor by which the selected reference clock is divided before input to the Counter Register increment logic. The two bit number represents the power of 10 by which the reference clock will be divided.

The actual period of the HRC Overflow pulses is determined by the selected clock, the Clock Scale Factor, and the contents of the Reset Register. Figure 2.3 indicates the range of periods and resolution available to the user if the internal reference clock is selected.

Figure 2.2 - HRC Control Register Bit Assignments

BITS[1]	COMMENTS[2]
11	Overflow Enable Bit '0' (default) - The Overflow Flag will never be set '1' (HRCOVFL) - The Overflow Flag will be set by an overflow in the Counting Register.
12	End Enable Bit '0' (default) - The End Flag will never be set. '1' (HRCEND) - The End Flag will be set by an external end pulse, or by a P pulse to the HRC.
13	Refernce Clock Source Bit '0' (default) - The internal 1MHz clock is selected. '1' (HRCEXT) - The External Reference Clock input is selected
14-15	Clock Scale Factor - Specifies the power of ten to divide the selected reference clock by. '00' - Divide by 1 '01' - Divide by 10 '10' - Divide by 100 '11' - Divide by 1000

1 All unused bits should be set to zero.

2 Mnemonics for the bit patterns are defined in Appendix A.

CONTROL REGISTER BITS 14,15	RESOLUTION (SEC.)	SELECTED COUNTING FREQUENCY (Hz.)	HRC OVERFLOW RANGE	
			PERIOD (SEC.)	FREQUENCY (Hz.)
'00'	1 micro	1 M	1 micro- 0.065536	1 M - 15.2588
'01'	10 micro	100 K	10 micro- 0.65536	100 K - 1.52588
'10'	100 micro	10 K	100 micro- 6.5536	10 K - 0.152588
'11'	1 milli	1 K	1 milli- 65.536	1 K - 0.0152588

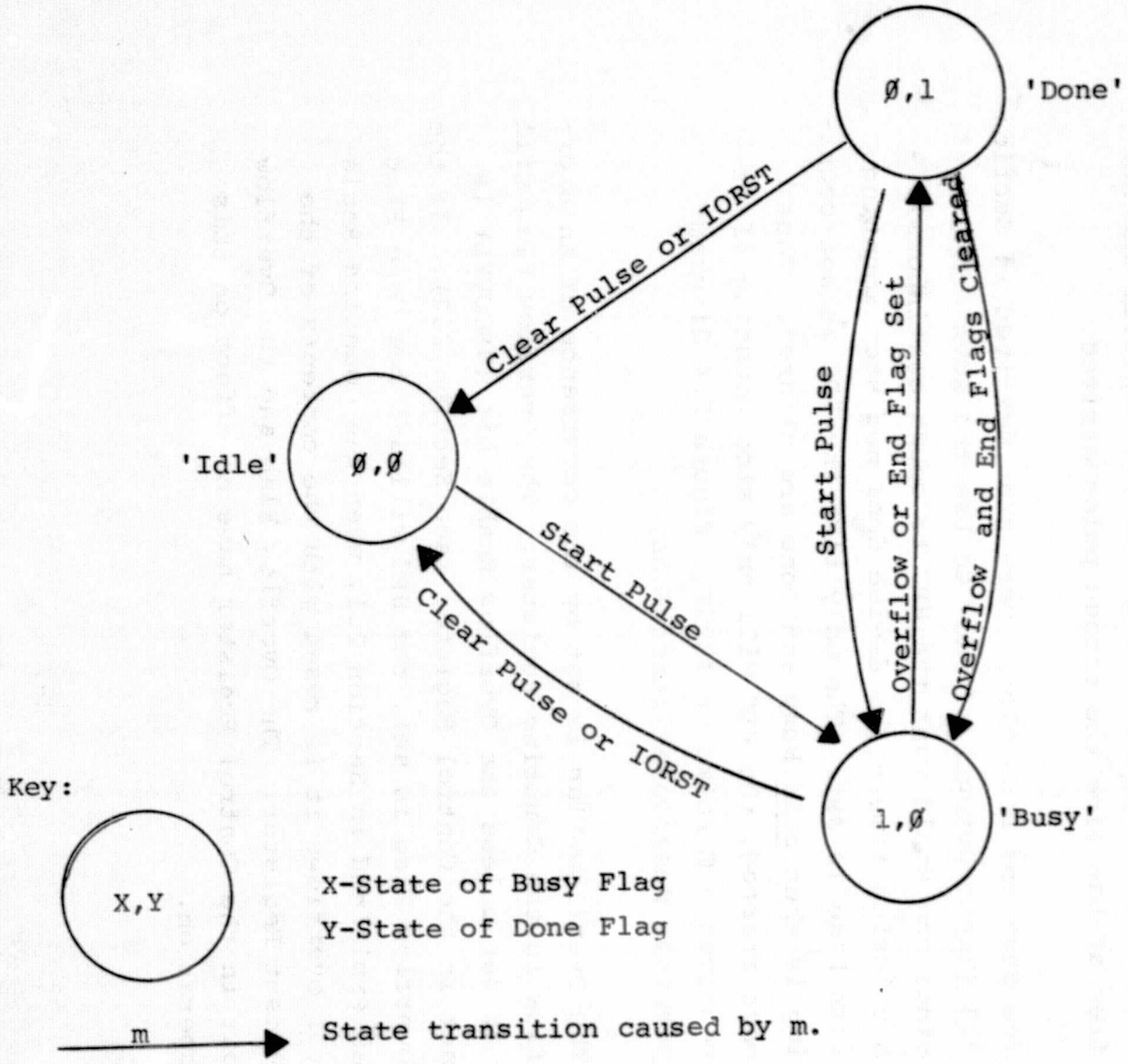
Figure 2.3 - HRC Timing Range for Internal 1 M Hz. Clock

2.3 Device Status Flags

In addition to the usual BUSY, DONE and INTERRUPT Disable status flags; the HRC contains an Overflow Flag, End Flag, and Error Flag. The Overflow Flag indicates that the 16 bit Counter Register has overflowed, while the End Flag indicates that an external End Pulse has been received. The Error Flag indicates that two External End Pulses have been received too close together, and that the Output Register was not free at the time the second pulse arrived.

The Busy and Done flags operate as desribed in Section 2.3 (Input-Output) of "How to Use the NOVA Computers". Since one would want the HRC to count continuously - not losing time - the device does not stop counting when Done is set. The only time the HRC is not counting is when both Busy and Done are cleared. Thus once started, the HRC will only stop counting if it receives a C Pulse or IORST. Figure 2.4 diagrams possible Busy/Done Transitions.

The Overflow Flag is set by the occurrence of an overflow in the Counting Register. The Overflow Flag will not set unless the Overflow Enable bit (HRCOVFL) is set in the Control Register (see Section 2.2). If the Overflow Flag is set, the HRC will set the Done Flag. As explained in Section 2.1, when the Counting Register overflows it is reset with the contents of the Preset Register. The Overflow Flag and the Overflow bit in the Control Register have no effect on this operation.



Note:

The HRC will continue to count in both the 'Busy' and 'Done' State.

Figure 2.4 - Busy, Done State Transitions for HRC

The End Flag is set by the occurrence of an External End Pulse (or P pulse to the HRC). The End Flag will not be set unless the End Enable Bit (HRCEND) is set in the Control Register. If the End Flag is set, the Output Register contains the value of the Counting Register at the time of the External End Pulse. The Counting Register continues to count and the Output Register will resume tracking when the End Flag is cleared. The Overflow Flag is also frozen by the End Flag. The Overflow Flag is treated just like the Output Register by the HRC, so that the Overflow Flag is valid for measuring intervals that are longer than one full Counting Register cycle.

The status of Busy and Done can be interrogated as standard practice for any device. If Done is set, then either the Overflow or End Flag must be set. If the End Flag is set, then the Error Flag may also be set. The state of the Overflow, End, and Error Flag may be read with a DIB d, HRC instruction. The bit assignments are as follows:

BIT	FUNCTION
Ø	End Flag
1	Error Flag
15	Overflow Flag

Any of the flags may be individually cleared by executing a DOB s, HRC to the device with the corresponding bit set. The following instruction sequence would read the flags into ACØ then clear only those flags set at the time of the DIB:

```
DIB Ø, HRC      ;READ STATUS FLAGS
DOB Ø, HRC      ;CLEAR ANY THAT WERE SET
```

after executing this instruction sequence, ACØ holds the device status and all the device flags have been cleared. Done is cleared, and Busy is set again (provided an S pulse had started the device previously).

The HRC can request interrupt service and acknowledge the INTA instruction as described in Section 2.4 - Program Interrupt of How to Use the NOVA Computers. The HRC will acknowledge with the value of the mnemonic 'HRC'. The Interrupt Disable flip flop of the HRC is attached to bit 1 of the bus. This interrupt mask is defined as MSHRC in Appendix A.

2.4 Programming the HRC

The simplest mode of operation for the HRC is as a periodic pulse generator to serve as a clock for other devices in the Analog Interface. In this case, the programmer wishes to set up the HRC and let it run without further intervention from the program. It would simply clock the other devices and the program would service their requests directly.

Programming Example 2.1 is a subroutine which will set up the HRC for its simplest mode of operation. The listing is well commented and considered self-explanatory.

Programming Example 2.2 makes use of the interrupt facility to provide a time of day clock in two core locations. To make the problem a little more interesting, let us assume that the programmer wishes to keep track of sideral time rather than solar mean time. A sideral second is equal to 0.997270 solar seconds. If one sideral second is sufficient resolution, then we wish to increment the core counter every 997,270 microseconds. The internal crystal reference in the HRC is accurate to about 1 part in 10^4 so rounding this figure to 997300 microseconds (an error of 3 parts ir 10^5) will not significantly affect the clock accuracy. This value simplifies programming the problem since it requires a basic clock period of 100 microseconds, and the full count of clock cycles (9973) will "fit in" the counting register. The code to implement this clock is presented in Programming Exmaple 2.2. It is assumed that the user will provide an interrupt handler that will JSR to HRCINT when HRC interrupts.

```
0001  PE2.1
;          (28,633)          JBW      MAY 1,1973       PAGE 1
;
;>         PROGRAMING EXAMPLE 2.1
;          ROUTINE TO SETUP HRC

;          ROUTINE TO SETUP THE HIGH RESOLUTION CLOCK
;          FOR TIMING OTHER DEVICES IN THE ANALOG
;          INTERFACE.  USER CAN SPECIFY PERIODS FROM
;          100 MICROSECONDS TO 6.5536 SECONDS.  THIS
;          CORRESPONDS TO A FREQUENCY RANGE OF 10 KHZ
;          TO 0.152588 HZ.

;>         CALLING SEQUENCE:

;          JSR @HRCST        ;BRANCH TO ROUTINE
;          <NORMAL RETURN>   ;RETURN HERE, HRC IS SETUP
;                            ; AND RUNNING

;          <HRCST MUST APPEAR IN AN .EXTD STATEMENT>

;>         ARGUMENTS:

;          AC0 - CONTAINS THE REQUIRED CLOCK PERIOD IN
;                  UNITS OF 100 MICROSECONDS.

;>         ACCUMULATORS AT RETURN:

;          AC1 AND AC2 ARE RESTORED.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0002  PE2.1
                ;       (32,701)        JBW     MAY 1,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 2.1

                ;       MAIN BODY OF CODE

                        .TITLE PE2.1
                        .ENT HRCST      ;HRCST IS ONLY LINK TO PROGRAM

                        .ZREL           ;PUT TRANSFER VECTOR IN PAGE ZERO
00000-000000'HRCST:     .HRCST

                        .NREL           ;CODE CAN GO ANYWHERE
00000'060232 .HRCST:    NIOC HRC        ;MAKE SURE HRC DEAD

00001'100400            NEG 0,0         ;REALLY WANT NEG OF PERIOD
00002'061032            DOA 0,HRC       ;SETUP PRESET REGISTER

00003'020403            LDA 0,HRCMD     ;GET BITS FOR CONTROL REGISTER
00004'063132            DOCS 0,HRC      ;SETUP CONTROL REGISTER AND
                                        ; START CLOCK RUNNING
00005'001400            JMP 0,3         ;RETURN TO CALLER

00006'000002 HRCMD:     2               ;CONTROL REGISTER BITS FOR-
                                        ; - NO OVERFLOW ENABLE (DEFAULT)
                                        ; - NO END ENABLE (DEFAULT)
                                        ; - INTERNAL 1 MHZ CLOCK (DEFAULT)
                                        ; - SCALE FACTOR OF 100 (10 TO THE 2ND),
                                        ;   YIELDS 10 KHZ COUNTING CLOCK

                        .END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0001  PE2.2
                ;       (48,1250)       JBW     MAY 1,1973      PAGE 1
                ;
                ;       PROGRAMMING EXAMPLE 2.2
                ;       A SIDERIAL CORE CLOCK

                ;       THIS EXAMPLE PROVIDES THE BASIC CODE TO
                ;       MAINTAIN A DOUBLE WORD CORE CLOCK.  THE
                ;       CLOCK IS MAINTAINED IN SIDERIAL SECONDS,
                ;       AS OPPOSED TO SOLAR MEAN SECONDS.
                ;       THE SYSTEM MUST BE RUNNING WITH INTERRUPTS
                ;       ENABLED FOR THIS SOFTWARE TO PROPERLY
                ;       MAINTAIN THE TIME.

                ;       THE CODE IS DIVIDED INTO TWO SECTIONS,
                ;       THE INITIALIZATION CODE AND THE INTERRUPT
                ;       CODE.  THE INITIALIZATION CODE SETS UP THE
                ;       HRC FOR PROPER OPERATION.  THE INTERRUPT CODE
                ;       IS CALLED BY THE USER SUPPLIED INTERRUPT
                ;       HANDLER.  THIS CODE UPDATES THE CORE CLOCK
                ;       AND RESTARTS THE HRC.

                ;       THE DOUBLE WORD CORE CLOCK IS MAINTAINED
                ;       IN PAGE ZERO UNDER THE ENTRY POINTS
                ;       'SIDHO' AND 'SIDLO'.  SIDHO IS THE HIGH
                ;       ORDER PART OF THE CLOCK, WHILE SIDLO IS
                ;       THE LOW ORDER PART.  SIDLO IMMEDIATELY
                ;       FOLLOWS SIDHO IN CORE.  TO ACCESS THESE
                ;       LOCATIONS, THEY MUST BE DECLARED IN AN
                ;       '.EXTD'.


                ;INITIALIZATION CODE:

                ;>      CALLING SEQUENCE:

                ;       JSR @SETHRC     ;BRANCH TO ROUTINE
                ;       <NORMAL RETURN> ;  HRC SETUP AND RUNNING
                                        ;  SIDHO & SIDLO ZEROED

                ;       (SETHRC MUST BE DECLARED IN AN '.EXTD')


                ;>      ARGUMENTS: NONE

                ;>      RETURN: HRC SETUP AND RUNNING.

                ;>      ACCUMULATORS AT RETURN: AC1 AND AC2 ARE
                ;               RESTORED.
```

```
^ 0002  PE2.2
                ;       (27,521)        JBW     MAY 1,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 2.2

                ;INTERRUPT CODE:

                ;>      CALLING SEQUENCE:

                ;       JSR @HRCIN      ;BRANCH TO HRC INTERRUPT HANDLER
                ;       <NORMAL RETURN> ; CORE CLOCK NOW UPDATED

                ;       (HRCIN MUST BE DECLARED IN AN '.EXTD')

                ;>      ARGUMENTS: NONE

                ;>      RETURNS: AFTER RETURN, CORE CLOCK HAS BEEN
                ;               INCREMENTED BY ONE.

                ;>      ACCUMULATORS: AC3 IS DESTROYED BY THE CALL.

                ;>      ARGUMENTS: NONE

                ;>      RETURN: AFTER RETURN, CORE CLOCK HAS BEEN
                ;               INCREMENTED BY ONE.

                ;>      ACCUMULATORS AT RETURN: AC3 IS DESTROYED
                ;               BY THE CALL.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0003  PE2.2
;       (50,1549)       JBW     MAY 1,1973      PAGE 3
;
;       PROGRAMMING EXAMPLE 2.2


;SPECIAL CONSIDERATIONS:

;       THE USER MUST PROVIDE AN INTERRUPT HANDLER
;       WHICH WILL JSR TO '.HRCIN' .  THIS HANDLER
;       IS EXPECTED TO LEAVE INTERRUPTS DISABLED
;       AT THE TIME IT CALLS '.HRCIN' .  THE USER
;       MUST REENABLE INTERRUPTS AFTER THE RETURN
;       FROM '.HRCIN' .

;       IF INTERRUPTS FROM THE HRC ARE DISABLED
;       FOR TWO SIDERIAL SECONDS, THE CLOCK WILL
;       LOSE TIME.

;       SINCE '.HRCIN' MODIFIES 2 WORDS IN CORE, IT IS
;       POSSIBLE FOR A USER PROGRAM TO READ A FALSE
;       CLOCK VALUE UNLESS PRECAUTIONS ARE TAKEN.
;       THE PROBLEM CAN OCCURE IN THE FOLLOWING CODE
;       TO READ THE CORE CLOCK -

;       LDA 0,SIDHO     ;GET HIGH ORDER TIME
;       LDA 1,SIDLO     ;GET LOW ORDER TIME

;       IF C(SIDHO) = 0 AND C(SIDLO) = 177777
;       (THE TIME IS 0177777 SECONDS, DOUBLE PRECISION)
;       AT THE TIME THE LDA 0,SIDHO IS EXECUTED, THEN
;       AC0 = C(SIDHO) = 0.  LET US ASSUME THAT
;       BEFORE THE LDA 1,SIDLO IS EXECUTED
;       THE HRC INTERRUPTS; THUS .HRCIN
;       UPDATES SIDHO TO 1 AND SIDLO TO 0.  WHEN
;       A RETURN FROM THE INTERRUPT ALLOWS THE LDA 1,SIDLO
;       TO BE EXECUTED, IT WILL LOAD AC1 WITH 0.
;       AT THIS POINT THE USER HAS A TIME OF 0,
;       WHEN THE TRUE TIME IS 200000.
;       THE SIMPLEST WAY TO AVOID THIS PROBLEM IS TO
;       DISABLE INTERRUPTS WHEN READING THE CLOCK.
;       FOR EXAMPLE-

;       INTDS           ;DISABLE INTERRUPTS
;       LDA 0,SIDHO     ;GET HIGH ORDER TIME
;       LDA 1,SIDLO     ;GET LOW ORDER TIME
;       INTEN           ;ENABLE INTERRUPTS AGAIN

;       NOTE THAT IF THE USER ONLY READS THE LOW
;       ORDER TIME (SIDLO) NO DIFFICULTY EXISTS,
;       SINCE ONE INSTRUCTION IS ALL THAT IS REQUIRED.
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0004  PE2.2
                ;       (44,895)        JBW     MAY 1,1973      PAGE 4
                ;
                ;       PROGRAMMING EXAMPLE 2.2

                ;       INITIALIZATION CODE

                        .TITLE PE2.2
                        .ENT SIDHO,SIDLO,.SETHRC,.HRCIN

                        .ZREL                   ;PUT CLOCK IN PAGE ZERO
00000-000000 SIDHO:     0
00001-000000 SIDLO:     0
00002-000000' SETHRC:   .SETHR                  ;VECTOR TO PRG
00003-000013' HRCIN:    .HRCIN                  ;VEC TO INTER HANDLER


                ;       ROUTINE TO SETUP THE HRC
                        .NREL

00000' 060232 .SETHR:   NIOC HRC                ;CLEAR HRC
00001' 102400           SUB 0,0                 ;A ZERO FROM NOTHING
00002' 040001-          STA 0,SIDLO             ;RESET TIME
00003' 040000-          STA 0,SIDHO

00004' 020405           LDA 0,HRCCR             ;GET BITS FOR CONTROL REG
00005' 063032           DOC 0,HRC               ;  SETUP REG

00006' 020404           LDA 0,HRCPR             ;GET VALUE FOR PRESET REGISTER
00007' 061132           DOAS 0,HRC              ;  SET REGISTER & START

00010' 001400           JMP 0,3                 ;RETURN

00011' 000022 HRCCR:    HRCOVFL+2               ;CONTROL REGISTER BITS FOR-
                                                ;  - OVERFLOW ENABLE (HRCOVFL)
                                                ;  - NO END ENABLE (DEFAULT)
                                                ;  - INTERNAL 1 MHZ CLOCK
                                                ;    (DEFAULT)
                                                ;  - SCALE FACTOR OF 100
                                                ;    (10 TO THE 2ND) YIELDS 100
                                                ;    MICROSEC.  BASE PERIOD.

00012' 154413 HRCPR:    -9973.          ;SET PERIOD TO 997300 MICROSECONDS -
                                        ;  A SIDERIAL SECOND IS 997270
                                        ;  MICROSECONDS.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0005  PE2.2
                ;       (16,333)        JBW     MAY 1,1973      PAGE 5
                ;
                ;       PROGRAMMING EXAMPLE 2.2

                ;       INTERRUPT CODE

00013' 060132 .HRCIN:   NIOS HRC        ;RESTART HRC FOR NEXT TICK

00014' 010001-          ISZ SIDLO       ;INC LOW ORDER PART OF TIME
00015' 000402           JMP .+2         ; NOT ZERO - SKIP SIDHO INC
00016' 010000-          ISZ SIDHO       ;LOW ORDER WAS ZERO - INC
                                        ; HIGH ORDER PART NOW
00017' 001400           JMP 0,3         ;RETURN TO CALLER
00020' 001400           JMP 0,3         ; SIDHO CAN GO TO ZERO!

                        .END
```

SECTION 3

THE ANALOG INPUT

3.0 General Information

The Analog Input (mnemonic AIN) provides sixteen analog input channels. Any channel may be selected under program control and a ten-bit sample obtained. Input conversions may be initiated under direct program control, from the HRC overflow pulse or from an external control line.

Figure 3.1 provides a block diagram of the system. The front end of the device consists of various channel conditioners (discussed in Section 3.1) which drive the 16 channel multiplexer. The selected channel is sampled by the Sample-and-Hold which in turn drives the 10 bit analog-to-digital converter. The output of the analog-to-digital converter may be read in by the user. In addition to the analog section, a digital section consisting of the Channel Select Register, the Control Register, the Status Flags and the Control Logic is included.

The contents of the Channel Select Register determine which of the input channels is selected by the multiplexer. If the user selects scan mode, the contents of the Channel Select Register is automatically incremented after each conversion, so that the next sequential channel is selected.

The Control Register determines the mode of operation of the AIN. It allows the user to specify the source

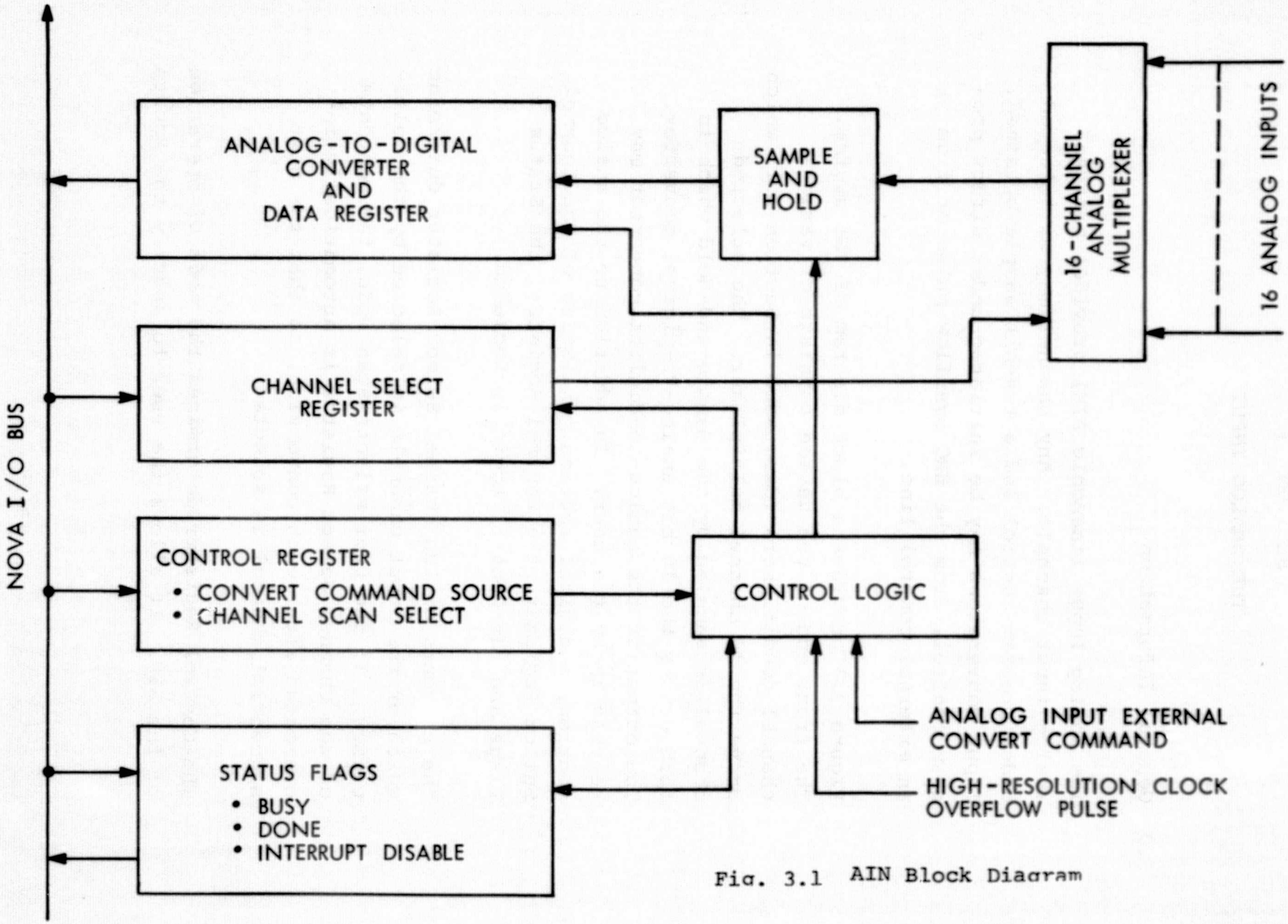


Fig. 3.1 AIN Block Diagram

of the command to initiate a new conversion cycle, as well as selection of scanning vs. non scanning mode. The user may select as the source of the conversion command an S pulse, the HRC Overflow Pulse, or the AIN External Sample Pulse. In addition, a P pulse will always initiate a conversion cycle.

The following events comprise a conversion cycle: First, the request for a conversion is noted by the hardware, when the multiplexer has selected the requested channel (as specified by the Channel Select Register) the conversion is initiated by sampling the output of the multiplexer. After the sample and hold has acquired the signal, the analog to digital converter begins the actual conversion to a digital representation. Since the sample and hold is maintaining a constant output, the multiplexer may be released to advance to a new channel. When the converter finishes, the AIN indicates that a valid conversion value is available to the user, by setting DONE.

3.1 Input Specifications

Sixteen channels are provided, eight of which are dedicated to special functions or conditioning. The other eight are direct inputs to the multiplexer (through protective circuitry).

Channels 0 thru 3 are connected to the Analog Knobs (see Figure 1.4) and are not available for signal input. Each Analog Knob is a precision 3 turn potentiometer. Full clockwise rotation corresponds to full scale positive input, while full counterclockwise rotation provides a full scale negative input.

Channels 4 through 7 are conditioned by preamplifiers before input to the Multiplexer. The gain of the preamplifiers may be adjusted from 0.5 to 10 at the front panel, and a DC offset of $\pm$ 5 volts may be added to the signal before scaling. The multiplexer full range input is $\pm$ 2 volts, so that signals in the range of $\pm$ .2 to $\pm$ 4 volts may be adjusted to match the multiplexer input range. Signal coupling, either DC or AC, is selectable and a ground position is provided for setting the zero point. At the unity gain setting, gain is down 3 dB at approximately 900 K Hz. The preamplifiers have an essentially constant gain-bandwidth product. With DC coupling, the response is flat down to DC, while with AC coupling the low frequency 3 dB point is approximately 1 Hz. The input impedance is 10 Megohms ($\pm$ 10%), provided the signal is within the $\pm$ 12 volt normal operating range.

A connector (see Appendix C) provides access to channels 8 thru 15 at the rear of the Analog Interface Chassis. These inputs are direct to the multiplexer inputs (through protective circuitry and require a signal

in the range of ± 2 volts. The input impedance is at least 10 megohms provided the input signal is within the ± 12 volt normal operating range.

All inputs are protected against excess voltage by an input clipping circuit that limits the input voltage to approximately ± 12 volts. If the input is within this range, the input impedance will be specified as above. If the input voltage should exceed this range, the input impedance will drop to no less than 3000 ohms as the protective circuitry activates. If the overvoltage is not destructive, the input will recover to normal operation after removal of the over voltage. The inputs will withstand a continuous input voltage of ± 30 volts; however, the user should take all reasonable precautions to restrict the input voltage to the normal operating range.

3.2 Data Register

The results of the latest conversion is contained in the Data Register. The contents of this register are valid only if the device is in the DONE state (see section 3.5). This register may be read with DIA d,AIN instruction. The format of the data is as follows:

Sign Bits		Nine low order bits	
Ø	6	7	15

Since only ten significant bits are generated by the analog to digital converter; the sign bit appears in bits 0-6 of the destination register while the nine low order bits appear in bits 7-15. The data is represented in the form of a signed twos complement number in the range -512 to 511 (177000_8 to 000777_8). This representation is the standard method of representing signed numbers in the NOVA, so that data is easily processed.

3.3 Channel Select Register

The Channel Select Register Specifies the input channel to be selected by the input multiplexer. This register is loaded with a DOA s,AIN instruction. The four bit channel address is taken from the bits 12-15 of the source register thus legal channel address are in the range of 0 to 15 (0_8 to 17_8). This register will remain fixed, unless the user has selected the scanning mode of operation. In this case, the register is incremented after each conversion. The register counts modulo 15 (i.e., 15_8, 16_8, 17_8, 00_8, 01_8).

3.4 Control Register

The Control Register determines the mode of operation of the device. It may be loaded with a DOC s,AIN instruction. The various control bits are assigned as shown in Figure 3.2.

Note that the device must be in the BUSY state before it will respond to the selected convert command. The S pulse is treated as a special case, which sets BUSY and requests a conversion at the same time.

In addition to the command sources specified in the Control Register, and P pulse will also initiate a conversion cycle. A P pulse will set Busy, and initiate a conversion cycle regardless of the mode selected by the Control Register. This feature is useful when the user wishes to sample a channel when the AINSP mode has not been selected.

Any unused bits in the Control Register should be set to zero.

BITS[1]	FUNCTION[2]
11	Selects channel scan mode of operation
	default (0) - The channel selected by the Channel Select Register remains unchanged.
	AINSCAN (1) - The Channel Select Register will be incremented during each conversion cycle, thus advancing thru the channels.
14-15	Selects the source of the command to initiate a conversion cycle.
	AINSP (00) - An S pulse will both set BUSY and start a conversion.
	AINHRC (01) - An High Resolution Clock Overflow Pulse will start a conversion provided the BUSY flag is set.
	-- (10) - Not used.
	AINEXT (11) - The external control signal, AIN EXT SAMP, will start a conversion provided the BUSY flag is set.

1. All unused bits should be set to zero.

2. Mnemonics for the bit patterns are defined in Appendix A (CBLDEFS).

Figure 3.2 - AIN Control Register Bit Assignments

3.5 Status Flags

The AIN contains standard Busy, Done, and Interrupt Disable flags as described in section 2.3, Input-Output of "How to Use the Nova Computers". Figure 3.3 diagrams the state transitions of the device. When the AIN is in IDLE state, it will not respond to the selected conversion command source. Either an S pulse or P pulse will take the device from IDLE state to BUSY state. The device will remain in BUSY state until a conversion cycle is completed or it is cleared by a C pulse or IORST. Entering BUSY state does not necesarily initiate a conversion cycle. The contents of the Control Register determines what signal initiates a conversion cycle. When a conversion cycle completes the device will go to the DONE state, thus indicating that valid data is available.

The AIN can be programmed for interrupt service as described in Section 2.4, Program Interrupt, of How to Use the NOVA Computers. The AIN will acknowledge the INTA instruction with the value of the mnemonic AIN. The Interrupt Disable flag is connected to bit Ø of the bus. This interrupt mask is defined as the mnemonic MSAIN in Appendix A.

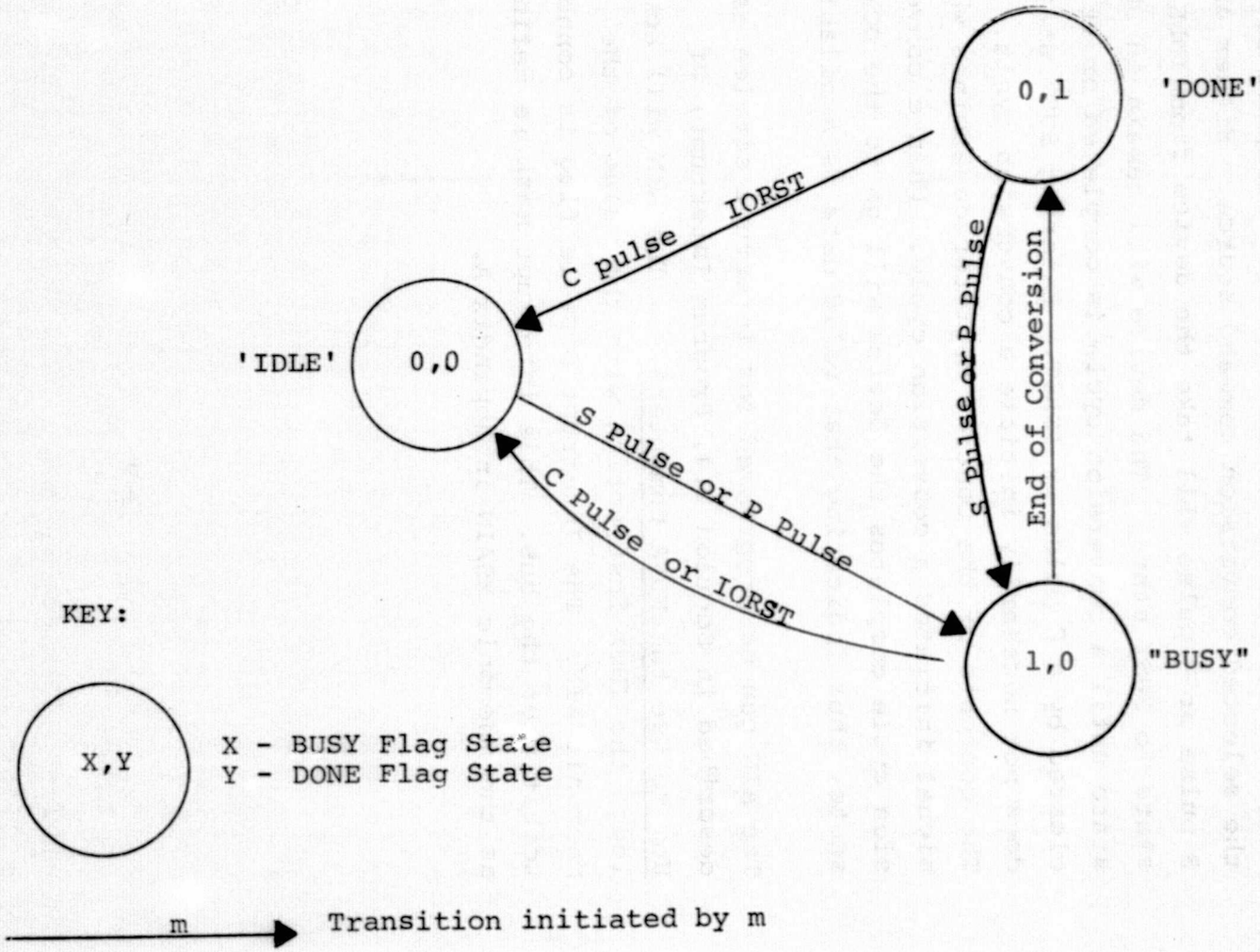


Figure 3.3 - AIN BUSY-DONE State Transitions

3.6 Analog Input Timing

The maximum time from the initiation of a sample conversion (the leading edge of a S pulse if AINSP mode is selected) until the data is ready to be read in is 8.75 microseconds. This assumes that the channel to be sampled was selected more than 5 microseconds before the request to initiate a conversion. If AINSCAN mode is enabled, the next sequential channel is selected while the old channel is being converted and is ready before the analog-to-digital converter finishes the current conversion. If the channel address and conversion request are given in the same instruction, then the channel settling delay (5 microseconds) must be added to the conversion time. Thus if the instruction,

```
DOAS 1,AIN      ;SELECT CHANNEL C(1) & CONVERT
```

is given, the time until the data is ready will be 13.75 microseconds from the leading edge of the DOA to the AIN.

This timing information is provided only to allow the programming of maximum sampling rates for the hardware. If the user does not need sampling rates approaching 50 kHz, he should always program with the I/O skip instructions to insure that the AIN is finished before reading the sample. This will assure compatible coding for the various NOVA processors which vary in instruction speed.

3.7 Programming the AIN

The simplest way to use the analog input is to sample signals on a single channel at some non-periodic rate. An example of this would be sampling a given analog input knob to determine a program parameter. In this case, the user must set up the device Control Register and select a channel. The following program segment demonstrates this:

```
SUB 0,0             ;CLEAR A CONVENIENT REGISTER
DOCC 0,AIN          ;LOAD THE CONTROL REG. HAVE SELECTED
                    ;NO SCAN AND S PULSE CONTROL
LDA 0,CHAN          ;LOAD CHANNEL NUMBER IN AC0
DOA 0,AIN           ;LOAD THE CHANNEL SELECT REGISTER
```

It is assumed that CHAN has been defined and contains the channel number to be sampled. The channel number would be between 0 and 3 if input from an analog knob were desired. Note that the user may sample the selected channel at any time by using the following sequence of instructions:

```
NIOS AIN            ;SET DEVICE BUSY AND START CONVERSION
SKPDN AIN           ;SKIP IF CONVERSION DONE
JMP .-1             ;LOOP WAITING FOR CONVERSION DONE
DIA 0,AIN           ;CONVERSION DONE, READ IN SAMPLE
```

The SKPDN is particularly important since it guarantees that valid data is present before it is read in. If for example, the DIA had immediately followed the NIOS, invalid data would be obtained.

In practice, the user is likely to require that the input signal be sampled at some constant rate.

He may accomplish this in several ways. One way is to use program loops for timing. This is very accurate on the NOVA since the CPU clock is crystal controlled. However, that this type of program is not directly transferrable between the various NOVA models, (i.e., NOVA, NOVA 800, NOVA 1200). A better solution is to use the High Resolution Clock provided as part of the Analog Interface. This clock, discussed in Section 2, provides the user with a wide range of possible sampling intervals. By setting AINHRC as the control mode, the programmer selects the High Resolution Clock as the source to initiate sample conversions. Programming Example 3.1 is the listing of a typical program using the High Resolution Clock to control the Analog Input. Since the program is heavily commented, only a few important points will be discussed. Note that the Control Register and Channel Select Register are loaded before setting the device BUSY. After the device is BUSY, the program simply waits for it to indicate the end of a conversion. The data is read in and the device is started again in the same instruction (DIAS Ø, AIN) to await the next High Resolution Clock Overflow.

It is the user's responsibility to assure that the sampling rate is not too high. In the previous example, the minimum clock interval was 100 microseconds. If, however, the clock had been set up to generate samples every 10 microseconds, samples would have been lost, since the NOVA processor cannot execute instructions fast enough to keep up with the clock.

Several channels of data may be being processed. In these cases, the user generally wants to minimize the time between samples (skew) on the various channels, as well as sample the channels periodically. The Scan Mode (bit 11 = 1 in the Control Register) is particularly suited for this provided the required channels are in sequence. If less than five are required, the data may be saved in the NOVA accumulators during the sequence. The following example demonstrates scanning three adjacent channels assuming that the initial channel has been preset in the Channel Selection Register and the Control Mode is set for High Resolution Clock or External Clock.

```
LP:  SKPDN AIN          ;WAIT FOR FIRST CONVERSION
     JMP .-1
     DIAP 0,AIN         ;READ IN FIRST SAMPLE AND START SECOND
     SKPDN AIN          ;WAIT FOR SECOND CONVERSION
     JMP .-1
     DIAP 1,AIN         ;READ IN SECOND SAMPLE AND START THIRD
     SKPDN AIN          ;WAIT FOR THIRD SAMPLE
     JMP .-1
     DIA 2,AIN          ;READ IN THIRD SAMPLE
     LDA 3, STRCH       ;GET STARTING CHANNEL IN SEQUENCE
     DOAS 3,AIN         ;RESET CHANNEL REGISTER FOR NEXT
                        ;SEQUENCE AND SET DEVICE BUSY TO
                        ;ACCEPT NEXT CONVERT COMMAND
     ...
     ...                ;PROGRAM CODE TO PROCESS DATA
     ...
     JMP LP             ;RETURN TO WAIT FOR FIRST SAMPLE OF
                        ;NEXT SEQUENCE
```

Again, the user must make sure that the time required to execute the loop does not exceed the interval at which sample conversions are initiated.

```
0001  PE3.1
                ;       (45,1098)       JBW     MAY 1,1973      PAGE 1
                ;
                ;       PROGRAMMING EXAMPLE 3.1
                ;       SAMPLING AN ANALOG SIGNAL

                ;       '.SAMP' IS A ROUTINE TO SAMPLE DATA FROM
                ;       A SINGLE CHANNEL AT A SPECIFED RATE.
                ;       '.SAMP' WILL STORE THE DATA IN A BUFFER
                ;       SUPPLIED BY THE USER.
                ;       THE HRC IS USED TO DETERMINE THE PERIOD
                ;       BETWEEN SAMPLES.

                ;>      CALLING SEQUENCE:

                ;       JSR @=.SAMP     ;BRANCH TO '.SAMP' USING A LOCAL
                ;                       ; ADDRESS CONSTANT.
                ;       <NORMAL RETURN> ;RETURN HERE -> BUFFER FILLED
                ;                       ; WITH SAMPLES.

                ;>      ARGUMENTS:

                ;       AC0 - CHANNEL # TO BE SAMPLED
                ;       AC1 - NUMBER OF POINTS TO BE SAMPLED
                ;       AC2 - ADDRESS OF 1ST WORD OF BUFFER
                ;       '.PERIOD' - LOCATION DEFINED IN '.SAMP' WITH
                ;               AN .ENT STATEMENT.  USER MUST DEFINE IN
                ;               AN '.EXTN' IF USED.  THIS LOCATION
                ;               SPECIFIES THE SAMPLE PERIOD IN UNITS
                ;               OF 100 MICROSECONDS.  INITIALIZED TO
                ;               10.  (1 KHZ) AT LOAD TIME.

                ;>      RETURNS:

                ;       WHEN THE NORMAL RETURN IS TAKEN, THE BUFFER
                ;       POINTED TO BY AC2 IS FILLED WITH C(AC1) SAMPLES
                ;       FROM CHANNEL C(AC0).

                ;>      ACCUMULATORS AFTER RETURN:

                ;       AC3 AND CARRY ARE DESTROYED, ALL OTHERS
                ;       RESTORED ON RETURN.

                ;>      SUBROUTINES USED:

                ;       CALLS '.HRCST' @ HRCST
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0002  PE3.1
                ;       (57,1004)       JBW     MAY 1,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 3.1 - CONT.

                ;       SIMPLE PROGRAM TO SAMPLE A FIXED # OF POINTS
                ;       FROM ONE CHANNEL.

                        .TITLE PE3.1
                        .ENT .SAMP,.PERIOD
                        .EXTD HRCST
                        .NREL

00000'040432 .SAMP:     STA 0,SV            ;SAVE ALL REGISTERS
00001'044432            STA 1,SV+1
00002'050432            STA 2,SV+2
00003'054432            STA 3,SV+3

                ;       SETUP AIN

00004'061231            DOAC 0,AIN          ;SET CHAN #, AND CLEAR AIN

00005'020424            LDA 0,AINMD         ;GET MODE 'AINHRC' FOR AIN
00006'063031            DOC 0,AIN           ; SETUP CONTROL REGISTER

                ;       SETUP THE HRC, USING .PERIOD

00007'020421            LDA 0,.PERIOD       ;GET PERIOD
00010'006001$           JSR @HRCST          ;GOTO SETUP ROUTINE


00011'124400            NEG 1,1             ;REALLY WANT NEG OF # OF POINTS

00012'060131            NIOS AIN            ;START AIN

                ;       MAIN SAMPLING LOOP NEXT

00013'063531 LOOP:      SKPBZ AIN           ;WAIT FOR SAMPLE
00014'000777            JMP .-1

00015'060531            DIAS 0,AIN          ;READ IN SAMPLE VALUE
                                            ; SET DEVICE BUSY FOR NEXT HRC
                                            ; OVERFLOW PULSE
00016'041000            STA 0,0,2           ;STORE DATA POINT

00017'151400            INC 2,2             ;UPDATE BUFFER POINTER

00020'125404            INC 1,1,SZR         ;MARK OFF A POINT DONE, CAN WE
                                            ; QUIT?
00021'000772            JMP LOOP            ; NO - GOTO WAIT FOR NEXT POINT

00022'060231            NIOC AIN            ; YES - CLEAR AIN AND HRC
00023'060232            NIOC HRC

00024'020406            LDA 0,SV            ;RESTORE REGISTERS & RETURN
00025'024406            LDA 1,SV+1
00026'030406            LDA 2,SV+2
00027'002406 .          JMP @SV+3           ;RETURN
```

```
^ 0003   PE3.1
                    ;       (12,234)        JBW     MAY 1,1973      PAGE 3
                    ;
                    ;       PROGRAMMING EXAMPLE 3.1 - CONT.

                    ;       CONSTANTS & SAVE AREA

00030' 000012 .PERIOD:          10.     ;DEFAULT PERIOD IS 1 MSEC.
00031' 000001 AINMD:    AINHRC          ;SELECT HRC AS SOURCE OF
                                        ; CONVERT COMMAND
       000004 SV:       .BLK 4          ;SAVE AREA FOR ACS

                        .END
```

SECTION 4

THE DUAL CHANNEL ANALOG OUTPUT

4.0 General Information

The dual channel analog output (mnemonic AOUT) consists of two 10 bit digital to analog converters and the associated control logic. The AOUT allows the user to generate two channels of analog signal output. Output conversions may be initiated under direct program control, from the HRC Overflow Pulse, or from an External Control Line.

Figure 4.1 is a block diagram of the AOUT system. The data path for each channel is from the program (via the NOVA bus) to the channel Data Register, from the Data Register to the Output Register, and finally to the 10 bit Digital-to-Analog converter. The user may directly load the Data Register. Only the AOUT Control Logic loads the Output Register. While the Data Registers are loaded independently, both Output Registers are loaded by the same signal, thus any changes in output are always synchronous (except for track made, see section 4.3). The Control Reigster selects the source of this signal.

The use of two buffers in each channel (double buffering) allows the user to set up the desired ouput value in advance of the actual time it is required without changing the previous output value.

Status falgs are provided to indicate the current status of the device and request program service.

The actual analog outputs are approximately $\pm$ 2 volts.

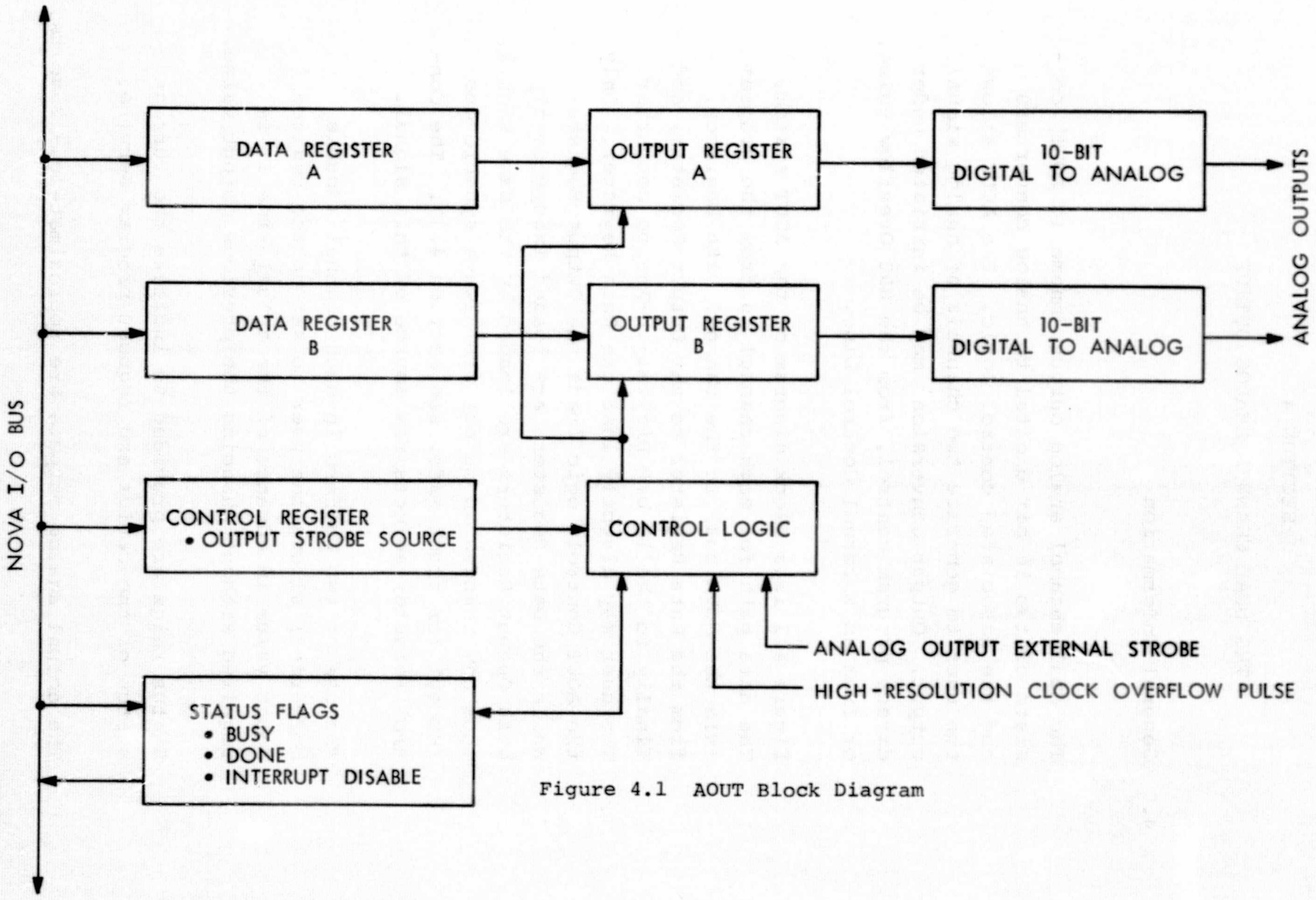


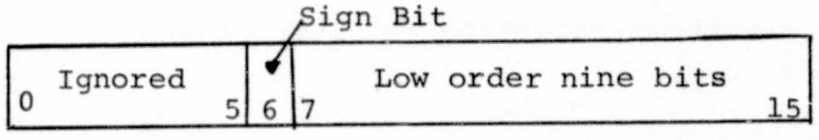
Figure 4.1 AOUT Block Diagram

4.1 Analog Outputs

Both analog output channels are available at a rear connector of the Analog Interface.* The output range is approximately $\pm$ 2 volts (the Least Significant Bit is nominally 4 millivolts). The minimum recommended output load is 1000 ohms.

4.2 Data Registers

A ten bit Data Register buffers the data for each D/A converter. The A channel register is loaded with a DOA s, AOUT and the B channel register with a DOB s, AOUT. Since only 10 bits are required by the D/A converters, the first six bits (0-5) are ignored. Bit 5 is treated as the sign bit in a twos complement representation. The format of the register is:



Note that signed numbers in the range 511 (000777_8) - 512 (177000_8) are mapped uniquely to an output voltage. This is compatible with the analog input converter and the internal representation of signed integers in the NOVA.

---

*See Appendix C for details.

4.2 Control Register

The Control Register specifies the mode of operation of the AOUT by selecting the source of the signal to initiate transfer of data from the Data Register to the Output Register. The contents of the Output Register determine the value of the analog output. The Control Reigster is loaded with a DOC s, AOUT instruction. The bit assignments and modes of operation are outlined in Figure 4.2.

The user may select the S pulse, HRC Overflow Pulse, AOUT External Strobe, or tracking mode. Busy must be set at the time of occurrence of the HRC Overflow Pulse or AOUT External Strobe, for the device to respond. The S pulse sets Busy and initiates the conversion at the same time, while in the AOUTTRK mode the output Registers always track the Data Registers without regard to the Busy Flag.

FIGURE 4.2

AOUT Control Register Bit Assignments

Bits[1]	Function[2]
14-15	Selects source of signal to initiate transfer of data from Data Register to Output Register, thus causing the analog outputs to acquire a new value.
	AOUTSP (00) - An S pulse sets Busy, clears Done and initates the transfer.
	AOUTHRC (01) - An HRC Overflow Pulse initiates the transfer, provided BUSY is set.
	AOUTTRK (10) - The Output Registers track the Data Registers, thus any data loaded into the Data Registers is also loaded into the corresponding Output Register. Busy/Done state does not affect this mode.
	AOUTEXT (11) - The AOUT External Strobe signal initiates the transfer of data provided BUSY is set.

[1] All unused bits should be set to zero.

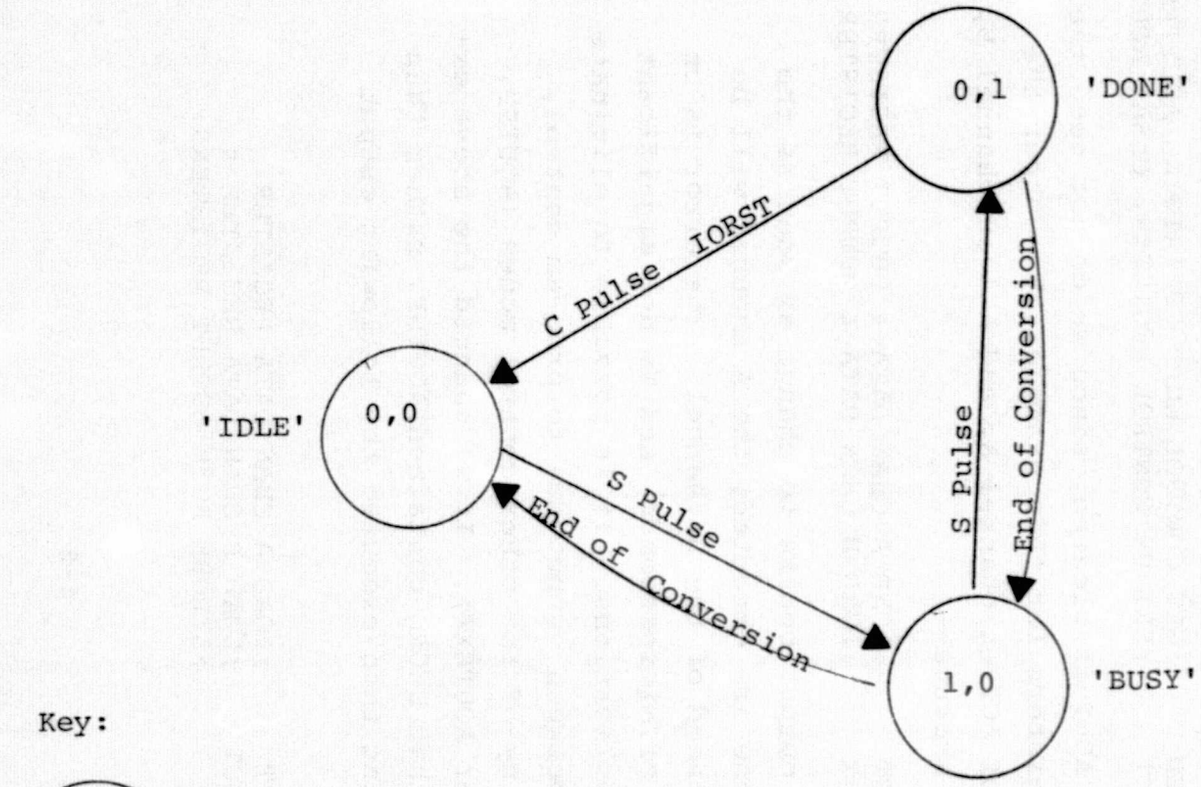
[2] Mnemonics are defined in Appendix A.

4.4 Status Flags

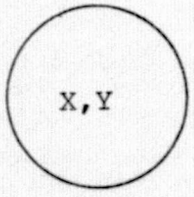
The AOUT contains standard Busy and Done Flags as described in Section 2.3, Input-Output of "How to Use the NOVA Computers". Figure 4.3 diagrams state transitions for the device. Note that a C pulse or IORST will always reset the device to IDLE mode. An S pulse will always place the device in BUSY state. With the exception of AOUTTRK (See Figure 4.2)mode, the device must be in Busy state before the analog ouptuts may change. If the Control Register has selected AOUTSP mode, the S pulse will simultaneously set Busy and initiate a conversion. The device will enter the DONE state after the analog output has acquired the new value.

In AOUTTRK mode, the user need not place the device in BUSY state to change analog values. If he does place the device in BUSY state, then it will never enter DONE state.

THE AOUT can be programmed for interrupt service as described in Section 2.4, Program Interrupt of How to Use the NOVA Computers. The AOUT will acknowledge the INTA instruction with the value of the mnemonic AOUT. The Interrupt Disable flag is connected to Bit 2 of the bus. This interrupt mask is defined as the mnemonic MSAOUT in Appendix A.



Key:



X - State of Busy
Y - State of Done



Transition initiated by m

Figure 4.3 - AOUT BUSY-DONE State Transitions

4.5 Programming the AOUT

The simplest mode of operation for the AOUT is AOUTTRK mode. In this mode, the output of a channel changes as soon as the instruction is executed. To set the AOUT for this mode of operation, the following sequence of instructions would be executed,

```
LDA Ø,CWORD      ;GET CONTROL WORD CONTAINING AOUTTRK
DOCC Ø,AOUT      ;SET UP CONTROL REGISTER, CLEAR AOUT
```

Note that AOUT is cleared, since we do not need the status registers for this example. To output the contents of ACØ on channel A, and AC1 on channel B, one would execute--

```
DOA Ø, AOUT      ;LOAD A CHAN DATA & OUTPUT REGISTER
DOB 1, AOUT      ;LOAD B CHAN DATA & OUTPUT REGISTER
```

Since the output begins to change as soon as the instructions are executed, the A channel will be slightly ahead of the B channel. The error is on the order of microseconds, and is not significant in many applications. It is possible to eliminate all skew between output due to program control, by using any of the other control modes (AOUTSP, AOUTHRC, or AOUTEXT). If we changed the above example so that CWORD contained AOUTSP, rather than AOUTTRK, and then executed the following output sequence,

```
DOA Ø, AOUT      ;LOAD A CHAN DATA REGISTER
DOBS 1, AOUT     ;LOAD B CHAN DATA REGISTER &
                 ;STROBE BOTH OUTPUT REGISTERS.
```

The actual analog output signals would not change until the S pulse on the DOBS occurred. At that time, both A and B would change to their new values, as specified by the most recent DOA and DOB instructions. If the user is processing or synthesizing analog signals, the output will have to be periodic. The HRC may be used to initiate output conversions (AOUTHRC mode). Programming Example 4.1 demonstrates an application of the AOUTHRC mode.

This example will output a buffer of data at a fixed rate and is essentially the complement of Programming Example 3.1. The example is well-commented and it is suggested the reader examine it carefully.

```
0001  PE4.1
                ;       (44,1185)       JBW     MAY 1,1973      PAGE 1
                ;
                ;       PROGRAMMING EXAMPLE 4.1
                ;       GENERATING AN ANALOG OUTPUT

                ;       '.DAOUT' IS A ROUTINE TO OUTPUT DATA IN
                ;       A BUFFER.  THE DATA IS DIRECTED TO
                ;       CHANNEL A OF THE AOUT.  THE HRC IS USED
                ;       TO SET THE INTERVAL BETWEEN CONVERSIONS.

                ;       CALLING SEQUENCE:

                ;       JSR @=.DAOUT      ;BRANCH TO '.DAOUT' USING A
                ;                         ; LOCAL ADDRESS CONSTANT.
                ;       <NORMAL RETURN>   ;RETURN HERE -> ALL DATA IN
                ;                         ; BUFFER HAS BEEN OUTPUT TO
                ;                         ; THE AOUT.

                ;       THE AOUT MAY BE BUSY AT THE TIME OF THE CALL,
                ;       '.DAOUT' WILL WAIT FOR PREVIOUS DATA POINT
                ;       TO BE CLOCKED OUT BEFORE PROCEEDING.

                ;>      ARGUMENTS:

                ;       AC0 - NUMBER OF POINTS TO BE OUTPUT
                ;       AC2 - ADDRESS OF 1ST WORD IN BUFFER OF DATA
                ;       '.DAP' - LOCATION DEFINED IN '.DAOUT' WITH AN
                ;                .ENT STATEMENT.  CONTENTS OF THIS
                ;                LOCATION SPECIFIES THE OUTPUT PERIOD
                ;                IN UNITS OF 100. MICROSEC.
                ;                INITIALIZED TO 10. AT LOAD TIME.

                ;>      RETURN:

                ;       AFTER A RETURN IS TAKEN, THE DATA POINTS
                ;       IN THE BUFFER POINTED TO BY AC2 HAVE BEEN
                ;       OUTPUT TO THE 'A' CHANNEL.  THE AOUT MAY
                ;       STILL BE BUSY AT RETURN TIME, AND THE USER
                ;       SHOULD SKIP ON THE BUSY FLAG BEFORE TAKING
                ;       ANY ACTION ON THE AOUT.

                ;>      ACCUMULATORS AT RETURN:

                ;       AC3 AND CARRY DESTROYED.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                ;       (54,1147)       JBW     MAY 1,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 4.1 - CONT.

                ;       PROGRAM TO OUTPUT DATA TO CHANNEL 'A'

                        .TITLE PE4.1
                        .ENT .DAOUT,.DAP
                        .EXTD HRCST
                        .NREL

00000' 040430 .DAOUT:   STA 0,SV        ;SAVE ALL REGISTERS
00001' 044430           STA 1,SV+1
00002' 050430           STA 2,SV+2
00003' 054430           STA 3,SV+3

                ;       WAIT TILL NOT BUSY, THEN SET UP THE HRC

00004' 063533           SKPBZ AOUT      ;WAIT FOR ANY PREVIOUS
00005' 000401           JMP .+1         ; CONVERSION

00006' 020420           LDA 0,.DAP      ;GET CLOCK PERIOD
00007' 006001$          JSR @HRCST      ; CALL ROUTINE TO SETUP HRC

00010' 034417           LDA 3,MODE      ;GET HRC MODE WORD FOR CONTROL
00011' 077233           DOCC 3,AOUT     ; SETUP AOUT CONTROL REGISTER


00012' 124401           NEG 1,1,SKP     ;WE REALLY WANT THE NEGATIVE OF
                                        ; THE # OF DATA POINTS. ALSO, SKIP
                                        ; 1ST ADDRESS ADVANCE
00013' 151400 LOOP:     INC 2,2         ;ADVANCE TO NEXT DATA ITEM

00014' 021000           LDA 0,0,2       ;PICK UP THE DATA
00015' 063533           SKPBZ AOUT      ;WAIT UNTIL NOT BUSY-> OK TO
00016' 000777           JMP .-1         ; RELOAD DATA REGISTER

00017' 061133           DOAS 0,AOUT     ;LOAD DATA REGISTER AND PLACE
                                        ; DEVICE IN BUSY STATE
00020' 125404           INC 1,1,SZR     ;MARK OFF ANOTHER DATA POINT IN
                                        ; DATA REGISTER
00021' 000772           JMP LOOP        ; NOT DONE YET -

                ;       ALL DATA ITEMS OUT, RETURN

00022' 020406           LDA 0,SV        ;RESTORE REGISTEPS
00023' 024406           LDA 1,SV+1
00024' 030406           LDA 2,SV+2
00025' 002406           JMP @SV+3       ;RETURN

00026' 000012 .DAP:     10.             ;PERIOD FOR HRC - DEFAULT 10.
00027' 000001 MODE:     AOUTHRC         ;SELECT HRC TO CLOCK DEVICE
       000004 SV:       .BLK 4          ;SAVE AREA FOR REGISTERS
                        .END
```

ORIGINAL PAGE IS OF POOR QUALITY

SECTION 5

THE DISPLAY ANALOG OUTPUT

5.0 General Information

The Display (mnemonic DISP) provides X and Y deflection signals for up to four display devices. The DISP consists of two ten-bit digital to analog converters, providing an addressable field of 1024 x 1024 points. Up to four devices may be attached to the output bus of the DISP. The same X and Y deflection signals are applied to each display device, however, the programmer may address the devices individually for beam intensification. Both oscilloscope displays and X-Y plotter are available.

Figure 5.1 presents a block diagram of the DISP logic. Note that it is similar to the AOUT logic. As in the AOUT, the Data Registers for each analog channel (X and Y) may be independently loaded with DOA (for the X channel) and DOB (for the Y channel) instructions. Output Registers for both channels are loaded at the same time by the Control Logic. The DAC's simply track any value in the Output Registers, converting it to the equivalent analog voltage. The output of the DAC's is summed with the analog signals X Character Deflection and Y Character Deflection. These signals are generated by the hardware character generator (described in Section 6, the reader may assume that they do not affect the value of the analog output for this section). The Control Register allows selection of the source of the command to plot a new point, selection of the output channels

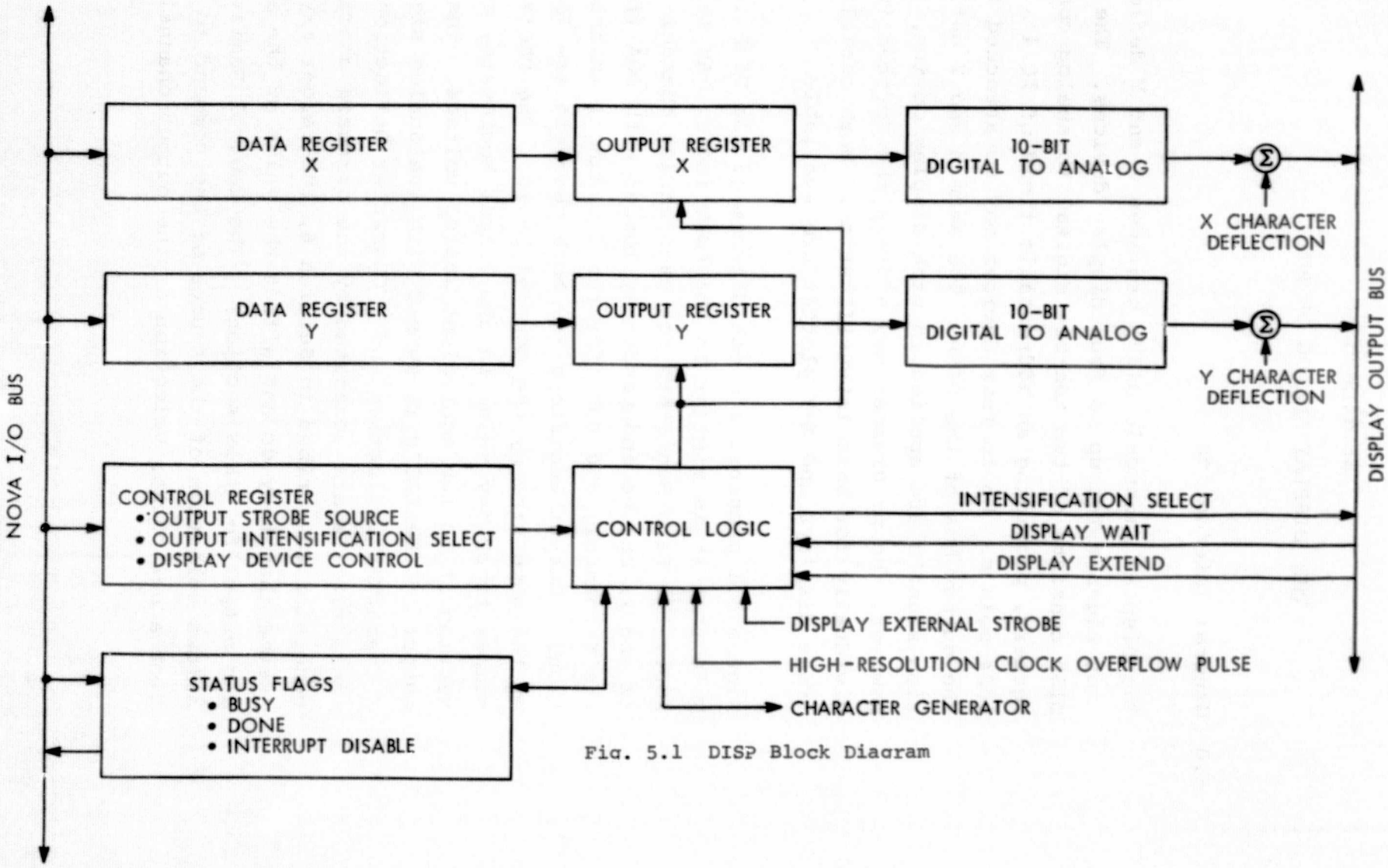


Fig. 5.1 DISP Block Diagram

which will be intensified, and specification of special device options. A plot cycle begins when the program senses (via the DISP Status Flags) the end of any previous display channel activity. At this time, the program would load the X and Y Data Registers with the position of the next point, and place the DISP in BUSY status. When the DISP Control Logic receives a command to plot the next point, it will load the Output Registers with the contents of the corresponding X and Y Data Registers, thus the analog output of the DAC's will change. When the X and Y signals have settled to the new position, a plot command is sent to every display output device selected in the Control Register. After a preset time, the DISP Control Logic will indicate that the plot cycle is complete by clearing the BUSY flag and setting the DONE flag. If a display device is not able to respond to the plot command in the allotted time, it will activate the signal DISP EXTEND. This will cause the Control Logic to hold off clearing BUSY and setting DONE, thus extending the length of the cycle.

In addition to extending the length of a plot cycle in progress, a display output device may indicate that it is not ready to accept changes in output by activating the DISP WAIT signal. The DISP Control Logic will not change the contents of the Output Registers while this signal is active, thus preventing any change in the X and Y analog signals. If the Control Logic receives a request to load the Output Registers while DISP WAIT is active, it will honor the request as soon as DISP WAIT becomes non-active. Typical applications of the WAIT feature would be

raising and lowering the pen on a hardcopy plotter or preventing writing on a storage scope while it was erasing the screen.

This character generator also interfaces with the Control Logic of the DISP to allow it to control display output devices while writing characters.

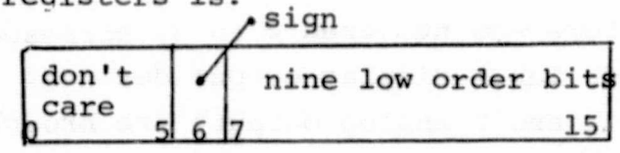
5.1 X-Y Outputs

Both X and Y analog output signals available at multiple connectors at the rear of the Analog Interface.* The output range is approximately $\pm$ 2 volts (the least significant bit is nominally 4 millivolts). The output connectors are numbered Ø to 3, corresponding to the four available display output devices. In addition, the X and Y analog outputs are brought out in the AOUT connector for convenience if the DISP is used as an additional pair of analog signal outputs. Each output connector is driven by an independent buffer. The recommended minimum output load is 1000 ohms.

* See Appendix C for details.

5.2 Data Registers

A ten bit Data Register buffers the data for each Digital to Analog converter. The X channel register is loaded with a DOA s, DISP while the Y channel register is loaded with a DOB s, DISP. The format of the registers is:



Since only ten bits are used by the DAC's, the first six bits (0-5) are ignored. Bit 6 is treated as the sign bit in a twos complement representation. Note that signed numbers in the range 511 (777_8) to -512 (177000_8) are mapped uniquely to an output voltage. This is compatible with the Analog input converter and the internal number representation of signed integers in the NOVA.

5.3 Display Field

The display field is a square grid of points with 1024 points on each side. As explained in the previous section, values of output voltage are represented by signed ten bit twos complement numbers. The display coordinate system is organized as shown in Figure 5.2. Note that the coordinate system origin is at the center of the screen, the X axis is the horizontal axis, and the Y axis is the vertical axis. There is one more negative number than positive number, due to the twos complement representation.

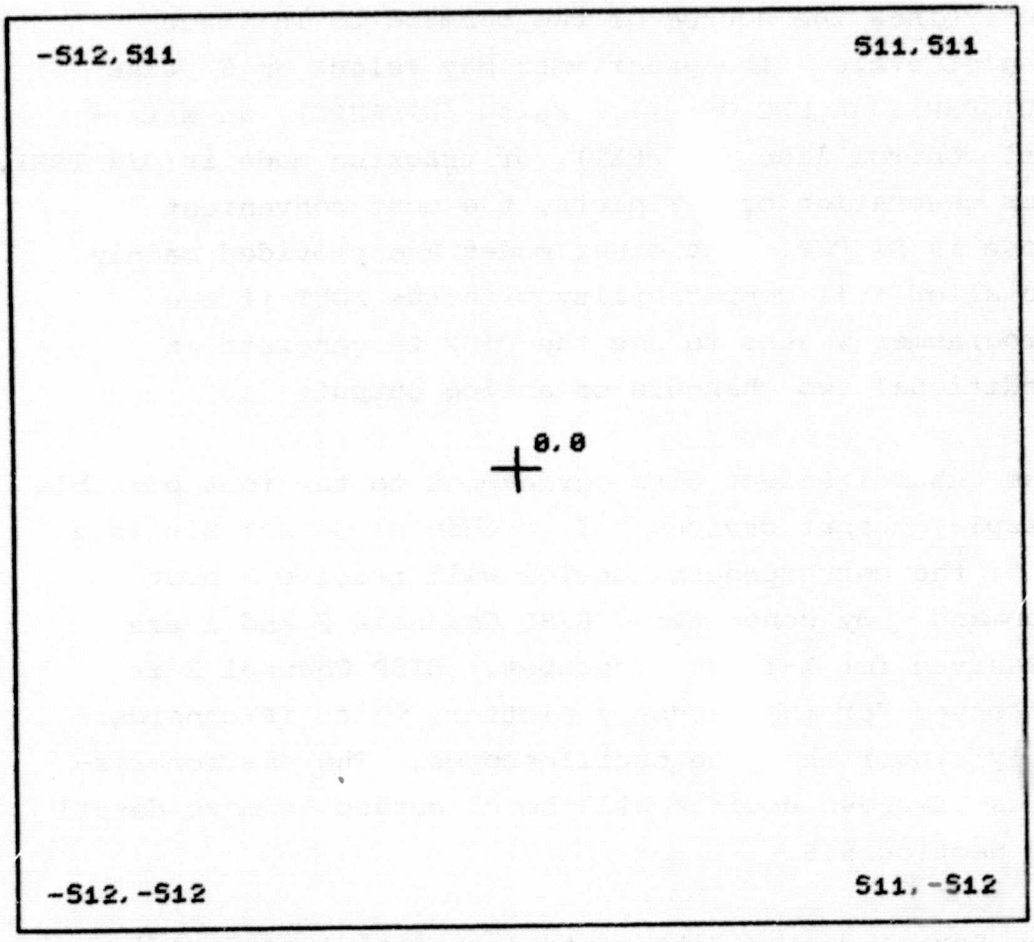


FIGURE 5.2 - DISP Coordinate System

5.4 Control Register

The control register selects the mode of operation of the DISP controller and the four possible display output devices. Figure 5.1 presents the organization of the Control Register. The mode bit determines the source of the command to initiate a plot cycle. The programmer may select an S pulse (DISPSP), an HRC Overflow Pulse (DISPHRC), an external control line (DISPEXT), or tracking mode is(DISPTRK). For generation of X-Y plots, the most convenient mode is DISPSP. The other modes are provided mainly to allow full compatability with the AOUT if the programmer wishes to use the DISP to generate an additional two channels of analog output.

The Channel Select Bits correspond to the four possible display output devices. If a Channel Select Bit is a "1", the corresponding device will receive a plot command. By convention, DISP Channels Ø and 1 are reserved for X-Y oscilloscopes. DISP Channel 2 is reserved for the hardcopy plotter, which is considerably slower than the oscilloscopes. The characteristics of these devices will be discussed in more detail in Section 5.6.

The Device Option Bits control special functions in the corresponding display output device. Their function is defined by the device attached to the channel, for example, the special function for the hardcopy plotter is pen position (Up or Down). An IORST will clear all Channel Select and Device Options Bits. A C pulse has no effect on the Control Register.

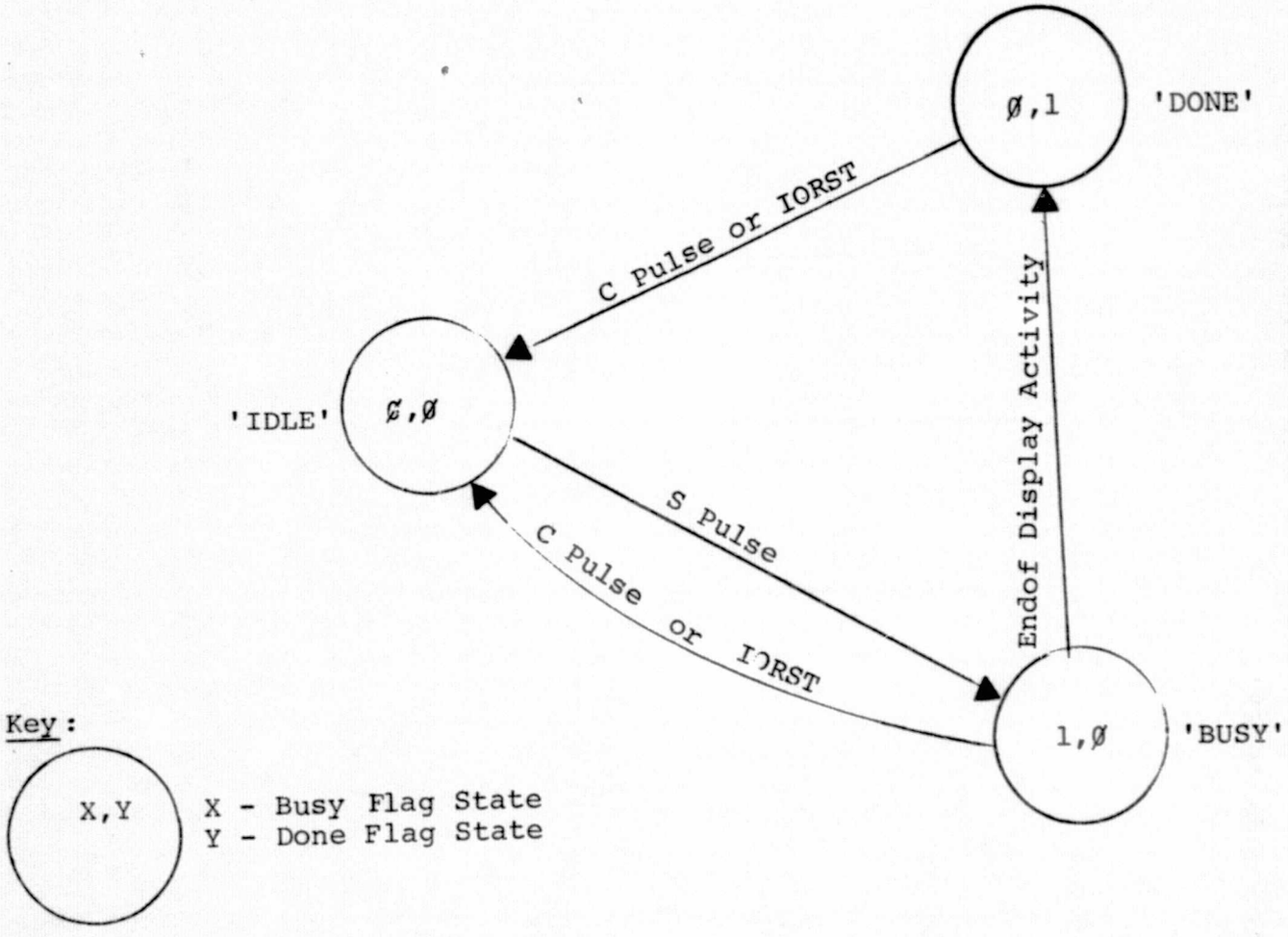
FIGURE 5.3 - DISP CONTROL REGISTER BIT ASSIGNMENTS

BITS	FUNCTION
Mode 14-15	Select source of signal to initiate a plot cycle. A plot cycle will transfer data from Data Registers to Output Registers, thus forcing the X and Y analog output to the new value.
	DISPSP (00) - An S pulse sets BUSY and transfers the plot cycle.
	DISPHRC (01) - An HRC Overflow Pulse initiates the plot cycle if BUSY is set.
	DISPTRK (10) - Out put Registers track Data Registers, thus the analog output is determined by the contents of the Data Register.
	DISPEXT (11) - The external control signal for the DISP (DISP EXT STB) initiates a conversion cycle provided BUSY is set.
Channel Select 4-7	A '1' specifies that this output device is to receive a command to plot a data point during the plot cycle.
	DISP0 (1B4)- Selects channel 0. By convention, an X-Y oscilloscope.
	DISP1 (1B5)- Selects channel 1. By convention, an X-Y oscilloscope.
	DISP2 (1B6)- Selects channel 2. By convention, an X-Y hardcopy plotter.
	DISP3 (1B7)- Selects channel 3.
Device Option 8-11	Enables a special function in the display output device associated with the channel.
	---- (1B8)- Enable function in channel 0.
	---- (1B9)- Enable function in channel 1.
	PENDWN (1B10)-Enable function in channel 2. By convention, this bit lowers the line drawing pen on the X-Y hardcopy plotter.
	---- (1B11)-Enable function in channel 3.

5.5 Status Flags

The DISP contains the standard BUSY, DONE, AND INTERRUPT DISABLE flags as described in How to Use The NOVA Computers, Section 2.3. With the exception of the DISPTRK mode, the DISP must be BUSY for the X and Y output signals to change. The DISP must be BUSY to initiate a plot cycle. Figure 5.3 presents the state transitions of the DISP logic.

If the mode is DISPSP, and S pulse will simultaneously place the DISP in BUSY state, and initiate a plot cycle. If the mode is not DISPSP, an S pulse will simply put the DISP in BUSY mode. The DISP will then wait for the selected control signal to initiate a plot cycle. At the end of the cycle, BUSY will be cleared and DONE set. Any requests to start a plot cycle received while BUSY is not set are ignored by the DISP, thus the program maintains full control over the DISP. A C pulse or IORST will clear BUSY and DONE. It is important that the programmer always test the Status Flags when programming the DISP as an X-Y display device. If these flags are tested, it guarantees that the program will never output data to the selected display output devices at a rate higher than they can accept. It is possible, on the NOVA 800, to output data to the X-Y oscilloscopes at a rate of 20 microseconds per point. If the hardcopy plotter is selected, its maximum rate is a point every 15 milliseconds. (Its minimum rate is a point every 1 second!) If the Status Flags are tested, the programmer need not worry about the special case of the plotter--the DISP will automatically extend the plot cycle from 20 microseconds to the actual time required to plot the point.



Note: All I/) control events are directed at the DISP or CHAR

FIGURE 5.3 - DISP State Transitions

Whenever DONE is set and INTERRUPT DISABLE is zero, the DISP will request interrupt service as specified in section 2.4, Program Interrupt, of "How to Use The NOVA Computers". The INTERRUPT DISABLE flag is attached to bit 3 (mnemonic MSDISP) on the NOVA data bus. The device will acknowledge the INTA instruction with the device code 'DISP'.

5.6 Standard Display Devices

The standard configuration of the system will include at least one small X-Y oscilloscope (8 x 10 cm or 4 x 5 inch screen) with a P7 phospher (long persistance). Optionally, the system will also include a large screen oscilloscope (8 x 10 inch) with a P7 phospher and an X-Y flatbed plotter (25 x 38cm). By convention, the following display output channels are assigned to these devices (if present):

Channel	Display Device
DISP0 (0)	Small screen oscilloscope
DISP1 (1)	Large screen oscilloscope
DISP2 (2)	X-Y flatbed plotter
DISP3 (3)	-----

Since both channels 0 and 1 are assigned to X-Y oscilloscopes, it is convenient to always select both of these channels unless the user wishes to maintain different displays on the two channels.

5.7 Programming the DISP

Let us say that we wish to plot some X-Y data on the two oscilloscope channels. The DISP Control Rigester would be set up with the following sequence--

```
        LDA Ø CWORD          ;GET CONTROL BITS FOR DISP
        DOCC Ø, DISP         ;SET CONTROL REGISTER
        .
        .
        . other code
        .
CWORD:  DISPØ+DISP1+DISPSP   ;THIS SELECTS CHAN's Ø & 1,
                             ;PUTS DISP IN S PULSE MODE
```

After executing the DOCC, the DISP would be ready to plot points in response to the following code. Assume that the X coordinate is in ACØ and the Y coordinate is in AC1.

```
        SKPBZ DISP           ;WAIT FOR END OF ANY PREVIOUS
        JMP .-1              ;PLOT CYCLE

        DOA Ø,DISP           ;LOAD X DATA REGISTER

        DOBS 1,DISP          ;LOAD Y DATA REGISTER
                             ;INITIATE A PLOT CYCLE
```

Note that the program waits for the BUSY flag to become "Ø", indicating that the last plot cycle has completed. Since after a Clear pulse, the BUSY flag is also Ø, this will work the first time (a SKPDN would not have worked the first time). After verifying that the previous plot cycle is completed, the X and Y Data Registers were loaded, and a new plot cycle initiated with the Start Pulse. Since the DISP X and Y outputs are double buffered, the following sequence would also work:

```
        DOA Ø,DISP           ;LOAD X DATA REGISTER

        DOB 1,DISP           ;LOAD Y DATA REGISTER

        SKPBZ DISP           ;WAIT FOR END OF PREVIOUS
        JMP .-1              ;CYCLE

        NIOS DISP            ;INITIATE A NEW PLOT CYCLE
```

The Output Registers are not loaded until the request to initiate a new plot cycle, so the DOA and DOB do not immediately change the X-Y output values. The above output code could also plot the X-Y data on the plotter if CWORD contained DISP2 also. The display interface would automatically wait on the plotter.

Programming Example 5.1 provides a simple routine to plot an array of data. The data is assumed to be in the range +511 and -512, and is plotted as a deflection in the Y axis, while the X axis is stepped for each point. This listing is well commented and will not be explained here. The reader should note the use of the signed comparisons defined in Appendix A.

```
0001  PE5.1
                ;       (57,1398)       JBW     MAY 1,1973      PAGE 1
                ;
                ;       PROGRAMMING EXAMPLE 5.1
                ;       PLOT A FUNCTION IN Y
                ;
                ;       ABSTRACT:
                ;
                ;       .PLTY - ROUTINE TO PLOT AN ARRAY OF DATA
                ;       VS THE X AXIS.  DATA IS ASSUMED TO BE
                ;       INTEGER IN THE RANGE +511. TO -512.
                ;       THE USER DEFINES THE REQUIRED PLOT BY
                ;       PASSING THE ADDRESS OF A CONTROL TABLE
                ;       WHICH DRIVES THIS ROUTINE.
                ;
                ;       CALLING SEQUENCE:
                ;
                ;       JSR @=.PLTY     ;BRANCH TO .PLTY THRU A
                ;                       ;LOCAL ADDRESS CONSTANT
                ;       <NORMAL RETURN> ;RETURN HERE WITH DATA PLOTED
                ;
                ;       THE CALLER MUST LOAD THE DISP CONTROL
                ;       REGISTER BEFORE CALLING .PLTY.
                ;
                ;       THE DISP MAY BE BUSY AT THE TIME OF THE CALL,
                ;       .PLTY WILL WAIT FOR ANY PREVIOUS DISP
                ;       ACTIVITY TO TERMINATE BEFORE PROCEDING.
                ;
                ;       ARGUMENTS:
                ;
                ;       AC2 POINTS TO PLOT CONTROL TABLE, WHICH IS
                ;       FORMATED AS DESCRIBED BELOW.
                ;
                ;       PLOT CONTROL TABLE -
                ;        REFERENCED BY FOLLOWING OFFSETS
                ;       IX - INITIAL X CORD OF DISPLAY
                ;       IY - INITIAL Y CORD.  FOR .PLTY, THIS
                ;            IS ADDED TO EACH Y DATA CORD.
                ;            AS IT IS PLOTED.
                ;       XS - X SPACING BETWEEN POINTS.
                ;       YS - Y SPACING (NOT USED)
                ;       DA - ADDRESS OF FIRST DATA ITEM.
                ;       LEN- LENGTH OF DATA BUFFER
                ;
                ;       OFFSET VALUES ARE DEFINED ON NEXT PAGE.
                ;
                ;
                ;       RETURNS:
                ;
                ;       AFTER NORMAL RETURN, ALL DATA HAS BEEN
                ;       OUTPUT TO THE DISPLAY.
                ;       THE DISPLAY MAY STILL BE BUSY.  IF THE
                ;       USER WANTS TO CLEAR THE DISPLAY, HE MUST
                ;       WAIT UNTIL IT IS DONE BEFORE CLEARING.
                ;
                ;       ACCUMULATORS:
                ;
                ;       ALL REGISTERS RESTORED EXCEPT AC3 AND CARRY.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0002  PE5.1
                ;       (17,291)        JBW     MAY 1,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 5.1 - CONT
                ;
                ;       DEFINITION OF CONTROL TABLE OFFSETS
                ;
                ;       USER MAY EDIT THIS PAGE INTO HIS SOURCE
                ;       TO DEFINE OFFSETS FOR SETTING UP THE
                ;       CONTROL TABLE.
                ;
        000000          .DUSR IX        =0
        000001          .DUSR IY        =1
        000002          .DUSR XS        =2
        000003          .DUSR YS        =3
        000004          .DUSR DA        =4
        000005          .DUSR LEN       =5
                ;
```

```
0003  PE5.1
                ;       (57,978)        JBW     MAY 1,1973      PAGE 3
                ;
                ;       PROGRAMMING EXAMPLE 5.1 - CONT.
                ;
                ;       .PLTY - THE REAL THING
                ;
                        .TITLE PE5.1
                        .ENT .PLTY
                        .NREL

00000' 040435 .PLTY:    STA 0,SV        ;SAVE ALL THE REGISTERS
00001' 044435           STA 1,SV+1
00002' 050435           STA 2,SV+2
00003' 054435           STA 3,SV+3

00004' 021000           LDA 0,IX,2      ;GET THE INTIAL X VALUE
00005' 040427           STA 0,XCORD     ; SAVE IT FOR THE INNER LOOP

00006' 025001           LDA 1,IY,2      ;GET Y OFFSET

00007' 021005           LDA 0,LEN,2     ;GET # OF POINTS
00010' 040423           STA 0,LPCT      ; SAVE IN LOOP COUNTER

00011' 035004           LDA 3,DA,2      ;GET ADDRESS OF DATA

00012' 031002           LDA 2,XS,2      ;LOAD X STEP SIZE


00013' 021400 LOOP:     LDA 0,0,3       ;GET DATUM
00014' 175400           INC 3,3         ;ADVANCE POINTER

00015' 123000           ADD 1,0         ;ADD ON OFFSET (IN AC1) TO Y

00016' 062034           DOB 0,DISP      ;LOAD DISPLAY Y BUFFER

00017' 020415           LDA 0,XCORD     ;GET CURRENT X CORD

00020' 063534           SKPBZ DISP      ;WAIT FOR END OF LAST POINT
00021' 000777           JMP .-1
00022' 061134           DOAS 0,DISP     ;LOAD X BUFFER & OUTPUT X & Y

00023' 143000           ADD 2,0         ;UPDATE X BY X STEP
00024' 040410           STA 0,XCORD

00025' 014406           DSZ LPCT        ;DONE YET?
00026' 000765           JMP LOOP        ; NO- GET NEXT POINT

00027' 020406           LDA 0,SV        ;YES- RESTORE REGISTERS
00030' 024406           LDA 1,SV+1
00031' 030406           LDA 2,SV+2
00032' 002406           JMP @SV+3       ;AND RETURN

       000001 LPCT:     .BLK 1          ;LOOP COUNTER
       000001 XCORD:    .BLK 1          ;TEMP X CORD
       000004 SV:       .BLK 4          ;4 WDS TO SAVE ACS

                        .END
```

ORIGINAL PAGE IS
OF POOR QUALITY

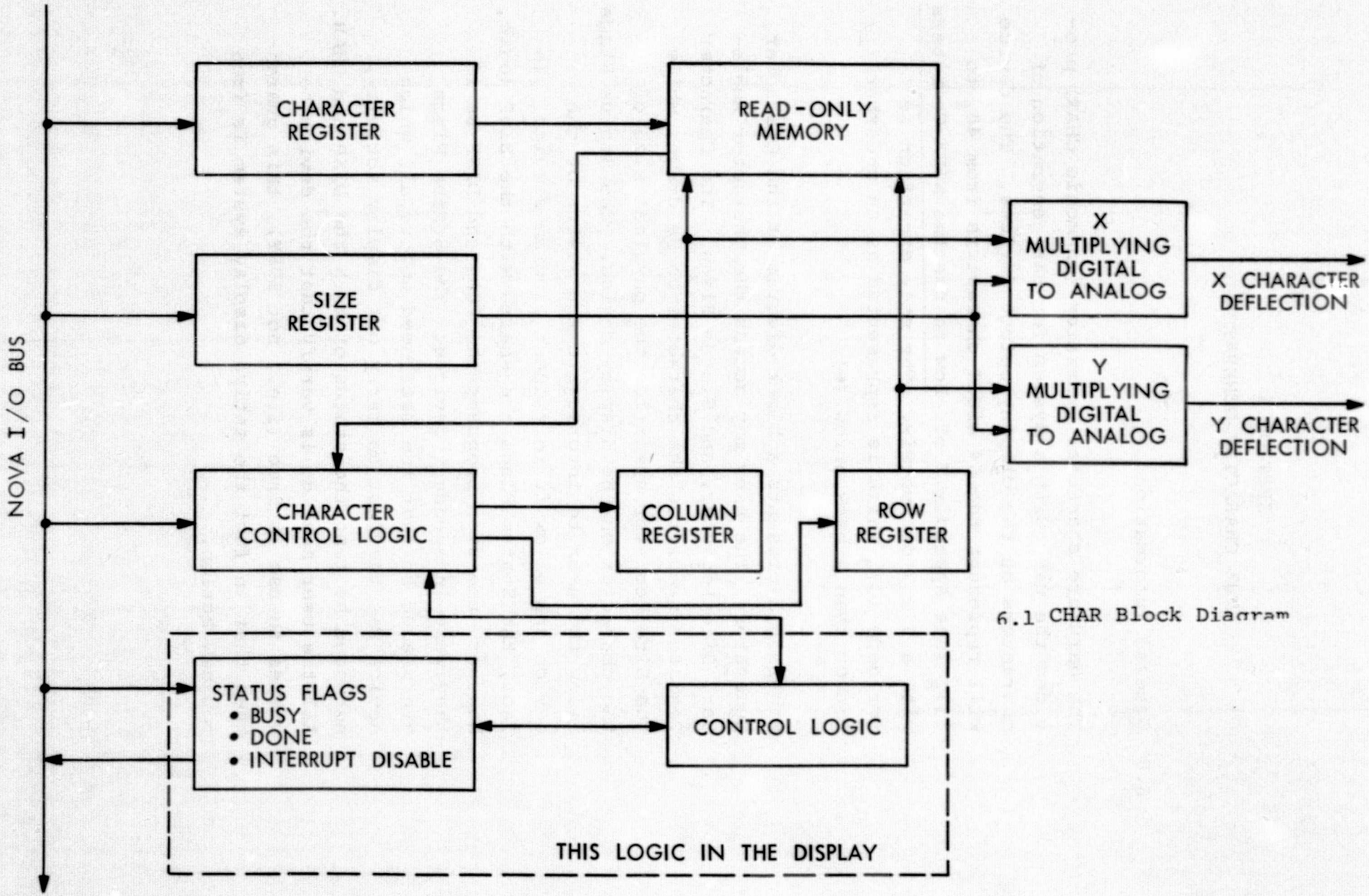
SECTION 6
THE CHARACTER GENERATOR

6.0 General Information

The hardware character generator (mnemonic CHAR) provides the user with a system for the generation of characters on the display output devices. The device will represent the 64 ASCII characters from 40_8 to 137_8 (see Appendix E of How to Use the NOVA Computers) with a 5 x 7 dot matrix. The only exception is for code 137_8 which is represented as an underbar (_) rather than a back arrow (←).

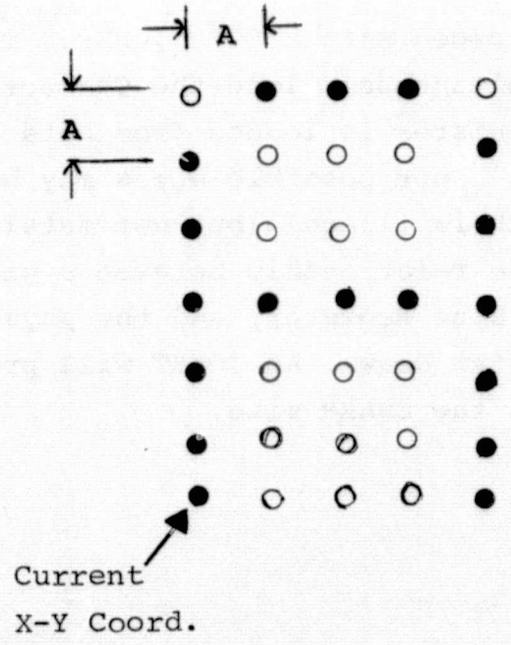
Figure 6.1 presents a block diagram of the Character Generator. The user may access the Character Register, Size Register, and Status Flags. The Character Register specifies the character to be drawn, while the Size Register specifies the physical size of the character on the display device. The Status Flags allow the user to interrogate the status of the device and allow it to request program service. In fact, the Status Flags are shared with the DISP logic, however, the user programs the CHAR and DISP as if they were independent devices. The Status Flags are shared so that the user need not recall which device was the last to carry out display activity, nor need he test the status of both the DISP and CHAR. All the user need do is verify that the device he wishes to use is free (i.e., not BUSY), this guarantees that in fact the entire display system is free for new action.

c.2



6.1 CHAR Block Diagram

The Column and Row Registers also specify an incremental X and incremental Y deflection signal that is added to the current X and Y coordinate specified by the Output Registers of the DISP. Figure 6.2 diagrams the dot pattern relative to the current X and Y position. The dots of Figure 6.2 are darkened to indicate the intensification pattern for the letter "A".



"A" is dot spacing and is set by the Size Register.

FIGURE 6.2 - CHAR Dot Matrix

6.1 Character Register

The Character Register may be loaded with a DOA s,CHAR or DOB s,CHAR. The DOB will also load the Size Register (section 6.2). The ASCII character code is taken from the low order six bits of the s register. Six bits provides for 64 unique characters. The characters that may be plotted are represented in Figure 6.3.

6.2 Size Register

The Size Register is loaded with a DOB s,CHAR. This instruction will simultaneously load the Character Register. The Size Register is loaded from bits 6 and 7 of the register. Four possible sizes may be specified, one of which is illegal (but not fatal). Figure 6.4 presents the relationship between a given character size in the Size Register, and the physical size of the character drawn. An IORST will preset the Size Register with the CHARM size.

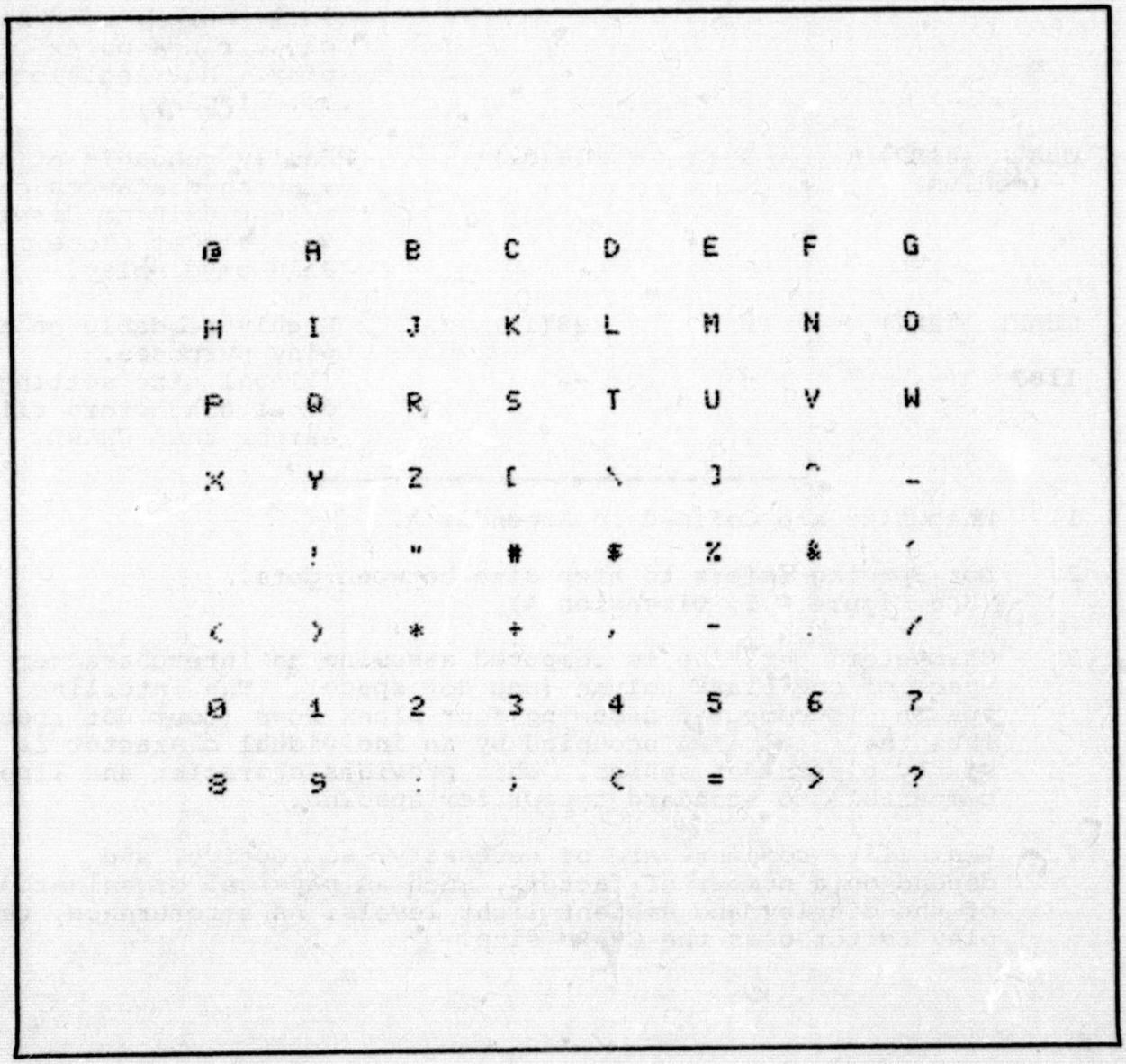


FIGURE 6.3 - CHAR Character Set

FIGURE 6.4

CHAR Character Size

Selected[1] Size	Dot[2] Spacing	Characters[3] Per Line (Lines on Display)	Legibility[4]
CHARS (00B7) (Small)	2	85(46)	Readable at close range on large screen display (8x10 in.) Barely readable at close range on 4x5 in display. Not legible on 8x10 cm. display.
CHARM (01B7) (medium)	3	56(31)	Easily readable at normal viewing distances on all except 8x10 cm display. Readable at close range on 8x10 cm display.
CHARL (10B7)	6	28(15)	Highly readable on all display purposes.
11B7	-	-	Illegal size setting, produces characters slightly larger than CHARL.

1. Mnemonics are defined in Appendix A.

2. Dot Spacing refers to step size between dots, (See Figure 6.2, Dimension A).

3. Characters per line is computed assuming an intercharacter space of one blank column (one dot space). The interline spacing is computed assuming four blank rows (four dot spaces). Thus the total area occupied by an individual character is six by eleven dot spaces. This provides character and line pitch comparable to standard typewriter spacing.

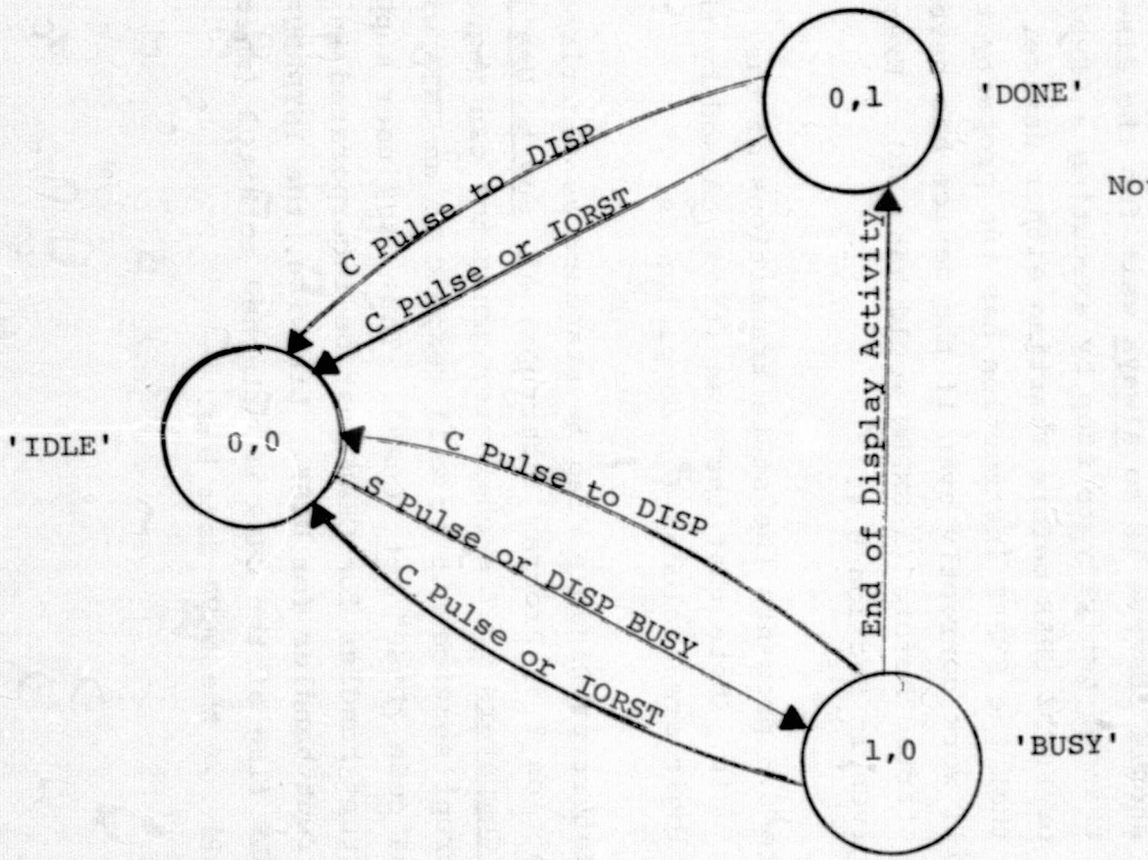
4. Legibility comments are of necessity, subjective, and depend on a number of factors, such as physical organization of the display and ambient light levels. As a reference, the Display Editor uses the CHARM size.

6.3 Status Flags

As was mentioned in Section 6.0, the Status Flags of the CHAR and DISP are shared. From the programmers point of view, the Status flags are as described in Section 2.3 of How to Use the NOVA Computers with the exception that the CHAR will indicate that it is BUSY if the DISP is using up the display channels. The only special precaution the user should observe is to always wait for the display channel activity to stop by executing a SKPBZ DISP or SKPBZ CHAR before starting either device. Note that the SKPBZ instruction has the property that it will work correctly even if the device has never been started before (a SKPDN would hang up). Examples are given in Section 6.4.

Figure 6.5 presents the state transitions of the CHAR logic. Note that the transitions caused by the DISP logic are included for completeness.

The Interrupt Disable Flag is standard as described in Section 2.4, Program Interrupt, of How to Use the NOVA Computers . The character generator can request interrupt service but it will respond to an INTA with device code "DISP" not "CHAR". Thus, any user supplied interrupt handler for CHAR must be incorporated in the interrupt handler for DISP. Likewise, the INTERRUPT DISABLE flag of the CHAR is attached to Bit 3 (mnemonic MSCHAR) of the NOVA data bus.



Note: End of display activity may be due to action initiated by CHAR on DISP

FIGURE 6.5 - CHAR BUSY-DONE State Transitions

6.4 Programming the CHAR

Since the CHAR draws the character at the current X-Y position, as determined by the DISP; the user is responsible for updating the X-Y position as required to output his text. The simplest method to do this is to place the display in DISPTRK mode, with the required display channels selected. Having done this, the following code would write the character in AC0 at the X-Y position determiend by the contents of AC1 and AC2.

```
SKPZ CHAR       ;WAIT FOR ANY PRECIOUS DISP OR CHAR
JMP .-1         ;ACTIVITY TO STOP

DOA 1,DISP      ;SET X POSITION
DOB 2,DISP      ;SET Y POSITION

DOAS 0,CHAR     ;LOAD CHARACTER REGISTER & START
                ;CHARACTER
```

Since the DISP was in track mode, the X and Y beam position began to change as soon as the DOA and DOB to the display were executed. When the DOAS to CHAR is executed, the X and Y outputs would not have settled to theri final position. The DISP logic will automatically hold off the character generator until the beam reaches the correct position. At this point it would allow the CHAR to actually begin plotting the character. Note that no S pulse was required to start the DISP, since it was in track mode and the CHAR controls intensification.

Programming Example 6.1 provides a routine which accepts a character and control table as input. It plots the character at the position indicated in the control table, then updates the X position in the control table so that the next character will appear to the right of the last one plotted. As it is called with successive characters, it generates a line of text across the display field.

It does not test for an end of screen or special control characters in its input, so the caller is responsible for maintaining line spacing.

```
0001  PE6.1
                ;       (45,1193)       JBW     MAY 2,1973      PAGE 1
                ;
                ;       PROGRAMMING EXAMPLE 6.1
                ;       DISPLAYING A CHARACTER
                ;
                ;       ABSTRACT:
                ;
                ;       .CHAR IS A ROUTINE WHICH WILL DISPLAY
                ;       A CHARACTER AND UPDATE A CONTROL TABLE.
                ;       THE CONTROL TABLE SPECIFIES THE POSITION
                ;       OF THE NEXT CHARACTER TO BE PLOTED.
                ;
                ;       CALLING SEQUENCE:
                ;
                ;       JSR @=.CHAR         ;BRANCH @ A LOCAL ADDRESS
                ;                           ; CONSTANT FOR .CHAR
                ;       <NORMAL RETURN>     ;RETURN HERE AFTER CHARACTER
                ;                           ; STARTED.
                ;
                ;       THE CALLER MUST LOAD 'DISPTRK' MODE INTO
                ;       THE DISPLAY CONTROL REGISTER BEFORE CALLING
                ;       .CHAR - AS WELL AS SELECTING THE DISPLAY
                ;       DEVICES.  THE CHARACTER SIZE REGISTER MUST
                ;       ALSO BE SETUP BY THE USER.
                ;
                ;       THE DISPLAY MAY BE ACTIVE (BUSY) AT THE TIME
                ;       OF THE CALL.  '.CHAR' WILL WAIT FOR ANY
                ;       PREVIOUS ACTIVITY TO TERMINATE BEFORE PROCEEDING.
                ;
                ;       ARGUMENTS:
                ;
                ;       AC0 HOLDS THE CHARACTER TO BE PLOTED IN THE
                ;       RIGHT BYTE.  ONLY BITS 10-15 ARE SIGNIFICANT
                ;       IN DETERMINING THE CHARACTER TO BE PLOTED.
                ;       AC2 HOLDS THE ADDRESS OF THE CHARACTER
                ;       CONTROL TABLE.
                ;
                ;       CHARACTER CONTROL TABLE-
                ;       CXC - X CORD OF NEXT CHARACTER
                ;       CYC - Y CORD OF NEXT CHARACTER
                ;       CSZ - CHARACTER SIZE, DOT SPACING
                ;             NOT HARDWARE SIZE.
                ;
                ;       CONTROL TABLE OFFSET VALUES DEFINED
                ;       LATER IN LISTING.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
;
;       RETURNS:
;
;       AFTER NORMAL RETURN, THE CHARACTER SPECIFIED
;       BY AC0 WILL HAVE BEEN OUTPUT TO THE CHARACTER
;       GENERATOR.  THE DISPLAY MAY STILL BE ACTIVE,
;       AND THE USER MUST WAIT UNTIL IT IS DONE
;       BEFORE HACKING IT.
;
;       THE CXC WILL HAVE BEEN UPDATED TO THE X CORD
;       OF THE NEXT CHARACTER - BASED ON THE VALUE OF
;       CSZ.  CYS IS NEVER MODIFIED BY '.CHAR'.
;
;       ACCUMULATORS:
;
;       AC2 IS RESTORED, ALL OTHERS DESTROYED.
;
;       SPECIAL CONSIDERATIONS:
;
;       THE USER MUST SPECIFY THE DOT SPACING FOR
;       THE CHARACTER SIZE HE HAS SELECTED.  THE
;       CORRESPONDENCE BETWEEN SIZE AND DOT
;       SPACING IS GIVEN BY THE FOLLOWING TABLE-
;
;       CHARS-  2
;       CHARM-  3
;       CHARL-  6
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                ;
                ;
                ;       PROGRAMMING EXAMPLE 6.1 - CONT.
                ;
                ;
                ;       DEFINITION OF CHARACTER CONTROL TABLE
                ;       OFFSETS.
                ;
                ;       THE USER MAY EDIT THIS PAGE INTO HIS
                ;       SOURCE TO DEFINE THE CONTROL TABLE
                ;       OFFSETS FOR INITIALIZATION OF THE TABLE.
                ;
000000                  .DUSR CXC       =0
000001                  .DUSR CYC       =1
000002                  .DUSR CSZ       =2
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0004  PE6.1
                ;       (43,821)        JBW     MAY 2,1973      PAGE 4
                ;
                ;       PROGRAMMING EXAMPLE 6.1 - CONT.

                ;       .CHAR - CHARCTER GENERATOR SUBROUTINE
                ;       USES HARDWARE CHARACTER GENERATOR.

                        .TITLE PE6.1
                        .NREL
                        .ENT .CHAR

00000'054420 .CHAR:     STA 3,SV3           ;SAVE RETURN ADDRESS

00001'063535            SKPBZ CHAR          ;WAIT FOR ANY PREVIOUS CHAR
00002'000777            JMP .-1             ; OR DISPLAY ACTIVITY

00003'035000            LDA 3,CXC,2         ;PICK UP X CORD
00004'075034            DOA 3,DISP          ; SET X CORD

00005'035001            LDA 3,CYC,2         ;PICK UP Y CORD
00006'076034            DOB 3,DISP          ; SET Y CORD - BEAM IS NOW
                                            ; MOVING TO LOWER LEFT HAND
                                            ; CORNER OF THE CHARACTER

00007'061135            DOAS 0,CHAR         ;START THE CHAR GENERATOR
                                            ; HARDWARE WILL HOLD IT
                                            ; UNTIL X & Y SETTLE

00010'021000            LDA 0,CXC,2         ;GET X CORD
00011'025002            LDA 1,CSZ,2         ;GET CHAR SIZE (I)
00012'135120            MOVZL 1,3           ; NOW 2I IN AC3
00013'165120            MOVZL 3,1           ; NOW 4I IN AC1
00014'137000            ADD 1,3             ; NOW 6I IN AC3

00015'163000            ADD 3,0             ;UPDATE X
00016'041000            STA 0,CXC,2         ; IN CONTROL TABLE

00017'002401            JMP @SV3 .          ;RETURN

        000001 SV3:     .BLK 1              ;SAVE RETURN ADDRESS


                        .END
```

SECTION 7

The General Purpose Register

7.0 General Information

The digital General Purpose Register (mnemonic GPR) consists of a separate 16 bit input register and and 16 bit output register and the standard Busy-Done network. The GPR can be used for I/O via the 16 data lines, as a buffer for user equipment connected to the NOVA, adn to test new I/O devices. The input register is a resettable binary counter, and the P pulse or a user supplied pulse will cause it to count.

Figure 7.1 provides a block diagram of the system. The GPR consists of the standard status flags (described in Section 2.3 of How to Use the Nova Computers, the Output Register, the Input Register and the Control Logic. Two connectors mounted on the rear panel provide access to the Input and Output registers and also to the status flags.

7.1 Output Register

The Output Register is a 16 bit latch followed by line drivers. The output lines (DATAOUTØ-DATAOUT15) are active low. If a user loads a '1' into a bit of the output register, then the corresponding output line will be low (about Ø.3 volts). Likewise, a 'Ø' loaded into the register is high (3.5 volts) at the output. The outputs are standard TTL levels capable of driving 30 TTL loads.

The Output Register is loaded with a DOA s,GPR instruction. In addition, the decoded DOB and DOC pulses are available at the rear connector. Appendix C provides detailed signal specifications and ocnnector pin assignments.

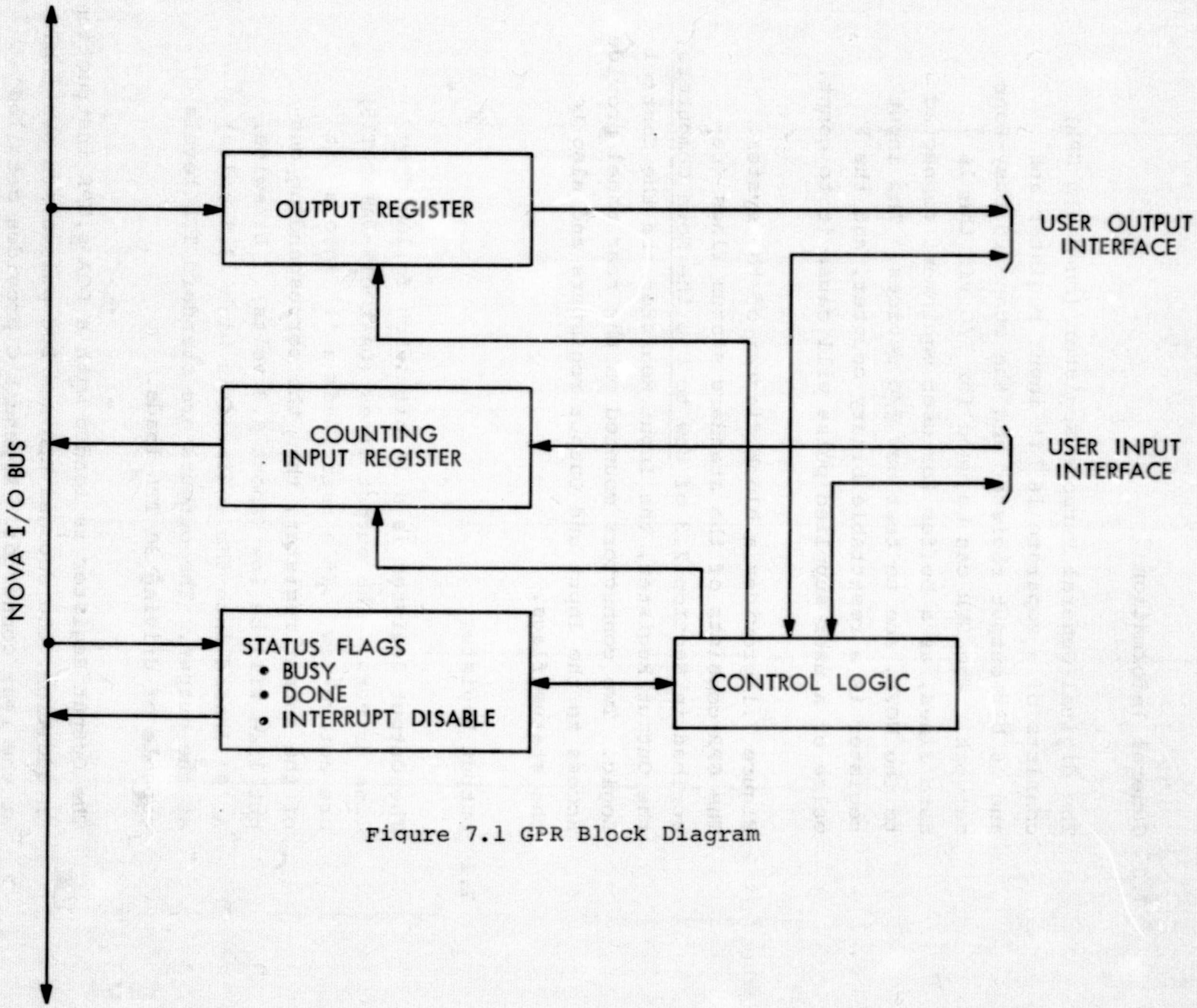


Figure 7.1 GPR Block Diagram

7.2 Input Register

The Input Register consists of a 16 bit resetable, parallel input counter. The input lines (DATAINØ-DATAIN15) are active low.

The Input Register may be read into an accumulator with either a DIA d, GPR or DIB d, GPR instruction. In addition, the DIC pulse is decoded and provided at the input connector. The DIA instruction loads the Input Register with the contents of the DATAIN lines, then loads the destination register with this value. The DIB instruction simply loads the current contents of the Input Register into the specified destination register. Thus, the DIA destroys any previous contents of the Input Register, while the DIB does not change the contents.

The contents of the Input Register can be incremented by a P pulse to the GPR or by an external USER COUNT signal. See Appendix C for details about the user signals and the connector pin assignments.

7.3 Status Flags

The GPR contains BUSY,DONE and INTERRUPT DISABLE flags as described in Section 2.3 Input-Output of How to Use the NOVA Computers . Figure 7.2 provides a diagram of the state transitions for the GPR.

The device may be programmed without the aid of the BUSY-DONE flags by simply issuing the appropriate input-output instructions. If the user places the GPR in BUSY state, by issuing an S pulse to the device, then the GPR will enter DONE state on the occurrance of a USER STROBE PULSE, USER DEVICE COMLETE, or DIA instruction. Thus, the user can set the device in BUSY state to await the completion of a task by the external user hardware.

The GPR can request interrupt service as described in section 2.4 - Program Interrupt of How to Use the NOVA computers . It functions in all respects as described in that section. The Interrupt Disable flag is attached to bit 5 (mnemonic MSGPR) of the NOVA data bus. The GPR will acknowledge the INTA instruction with the device code "GPR" (see Appendix A).

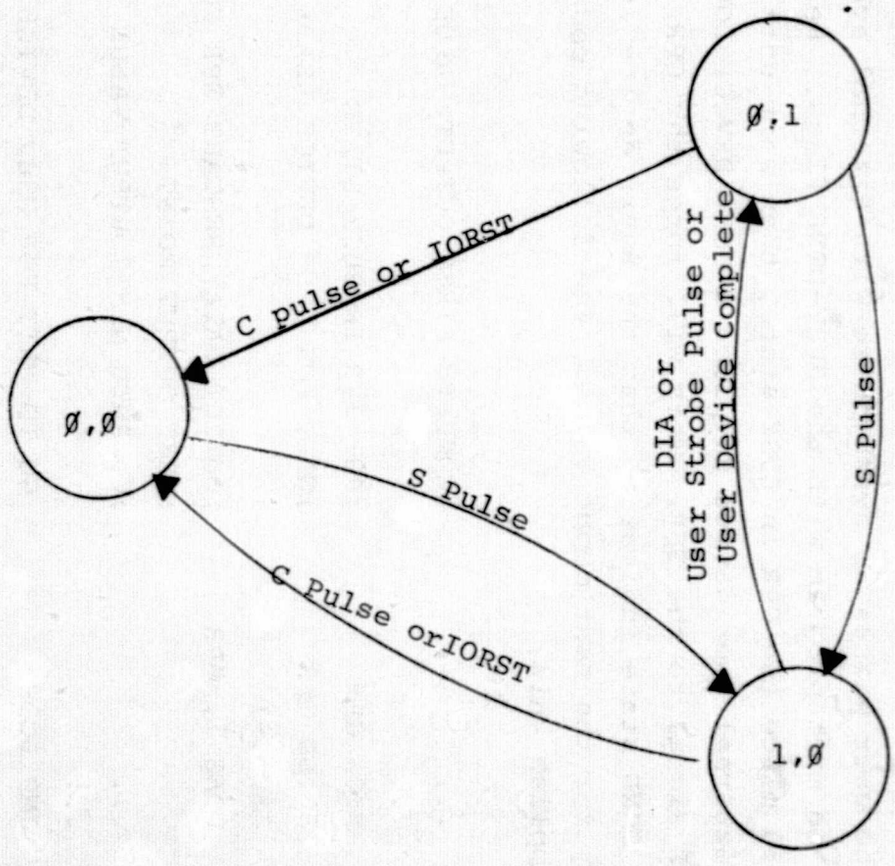


FIGURE 7.2 - GPR Busy-Done State Transitions

7.4 Programming the GPR

The simplest case of programming the GPR output occurs when the user wishes to simply output data without sychronizing with any external events. In this case, he may execute the instruction,

DOA s,GPR

to load the Output Register with C(s).

If the user wishes to synchronize with external events, then he must program with the BUSY-DONE flags. He should place the GPR in BUSY state, then wait until the external user device supplied a USER DEVICE COMPLETE signal to the GPR. This will force the GPR into DONE state indicating that the users device is ready for the next ouput value. The following code would accomplish this,

```
        ...             ;COMPUTE FIRST OUTPUT VALUE
        ...
        NIOS GPR        ;PUT GPR IN BUSY STATE
LOOP:   SKPBZ GPR       ;WAIT FOR USER DEVICE READY
        JMP .-1
        DOAS 1, GPR     ;OUTPUT DATA, RESTART GPR FOR
                        ;NEXT OUTPUT POINT
        ...             ;COMPUTE NEXT OUTPUT VALUE
        ...
        JMP LOOP        ;GO TO WAIT FOR USER DEVICE READY
```

As for output, the simplest case in programming the GPR for input operation occurs when the user need not synchronize with any external events. In this case, the user may simply use the instructions,

```
        DIA d,GPR               ;READ IN DATA LINES, DESTROYING OLD
                                ;CONTENTS OF INPUT REGISTER
```

or

```
        DIB s,GPR               ;READ IN INPUT REGISTER WITHOUT
                                ;DESTROYING CONTENTS.
```

If the user wishes to synchronize with external events, then he must use the BUSY-DONE flags. For example, assume that the user device is supplying USER COUNT PULSEs to the GPR input register. When it has completed the count for a particular run, then it supplies a USER DEVICE COMPLETE signal to indicate that the total count is available. The following code would handle this,

```
        ...             ;USER SETUP CODE
        ...
        NIOS GPR        ;SET GPR BUSY TO AWAIT COMPLETION
LOOP:   SKPDN GPR       ;WAIT TIL DONE
        JMP .-1
        DIBS 1,GPR      ;READ IN COUNT, RESTART DEVICE
        ...             ;PROCESS COUNT
        ...
        JMP LOOP        ;GOTO WAIT FOR NEXT COUNT
```

In this example, the user device would have to reset (REGRST signal) the Input Register each time it began a new run.

Programming example 7.1 is a program to decode a 16 key pushbutton keyboard. The program requires an interrupt handler and a routine which will set the HRC or RTC to interrupt at a rate of 1000 Hz. When the clock interrupts the data on the GPR lines is sampled and the program filters and decodes the keyboard. It is assumed that the keyboard is wired to ground the corresponding DATAIN line when a key is depressed. The data in lines are pulled up to

5 volts inside the GPR (see Appendix C) thus the line is high (a logic 'Ø' to the program) provided the key it is attached to is not depressed.

The problems of contact bounce in mechanical switches should not be overlooked when designing user interfaces to the GPR, or in programming handlers for the GPR Input Register. The programming example allows the use of simple switches by integrating the sampled switch position. This serves to eliminate switch bounce and noise problems. Alternatively a more complex switch interface could be used to eliminate any switch bounce before the GPR inputs.

```
0001  PE7.1
                ;       BG      5/2/73
                ;
                ;       PROGRAMMING EXAMPLE 7.1
                ;       GPR KEYBOARD INTERFACE
                ;
                ;
                ;       ABSTRACT:
                ;
                ;       THIS PROGRAM WILL DECODE KEYS STRUCK ON
                ;       A 16 KEY KEYBOARD.  THE FOLLOWING CODE
                ;       ASSUMES THAT AN INTERRUPT ROUTINE AND A
                ;       ROUTINE TO START EITHER THE RTC OR HRC TO
                ;       INTERRUPT AT A RATE OF 100 TO 1000 HZ  IS
                ;       PROVIDED.  WHEN A KEY IS RECEIVED, A USER
                ;       ROUTINE (PTKEY) IS CALLED TO STORE THE KEY.
                ;       A SIMPLE FILTER IS PROVIDED TO MAKE SURE THE
                ;       KEY IS DEPRESSED FOR A CERTAIN NUMBER OF
                ;       CLOCK CYCLES TO ELIMINATE NOISE SPIKES.
                ;
                ;       CALLING SEQUENCE:
                ;
                ;       JSR GTKEY       ;GET A KEY
                ;       [RETURN]
                ;
                ;       ARGUMENTS:
                ;
                ;       THE KEY STRUCK IS RETURNED TO A USER SUPPLIED
                ;       ROUTINE CALLED  PTKEY
                ;       WITH THE KEY (IN ASCII) IN AC0.
                ;
                ;
                ;       RETURNS:
                ;
                ;       GTKEY RETURNS AFTER INITIALIZING THE GPR.
                ;       THE USER ROUTINE PTKEY SHOULD RETURN TO
                ;       CALL+1 AFTER STORING THE KEY AND SETTING
                ;       A KEY RECEIVED FLAG.
                ;
                ;
                ;       ACCUMULATORS: AC3 IS DESTROYED
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0002  PE7.1
                ;       PROGRAMMING EXAMPLE 7.1

                        .TITL   PE7.1
                        .ENT    GTKEY,KEYINT
                        .EXTN   CLKST,PTKEY
                        .NREL

                ;THE FOLLOWING IS THE USERS CALL TO GTKEY
                ;AND THE USER'S PTKEY ROUTINE
                ;
                ;       SUB 0,0
                ;       STA 0,KRCVD     ;RESET KEY RECEIVED FLAG
                ;       JSR @.GTKEY     ;GET A KEY
                ;       ...
                ;       ...
                ;.GTKEY: GTKEY
                ;       ...
                ;       ...
                ;PTKEY: STA 0,KEY       ;STORE THE KEY THAT WAS STRUCK
                ;       SUBZL 0,0
                ;       STA 0,KRCVD     ;SET KEY RECEIVED FLAG
                ;       JMP 0,3         ;RETURN


                ;THE FOLLOWING IS THE GTKEY ROUTINE

00000'054406 GTKEY:  STA 3,GTRET     ;SAVE RETURN ADDRESS
00001'060236         NIOC GPR        ;CLEAR THE GPR
00002'020405         LDA 0,M5        ;SET THE KEY FILTER
00003'040406         STA 0,FKEY
00004'006404         JSR @.CLKST     ;JMP TO THE CLOCK START ROUTINE
00005'002401         JMP @GTRET

00006'000000 GTRET:  0
00007'177773 M5:     -5              ;FILTER CONSTANT
00010'177777 .CLKST: CLKST           ;USER CLOCK ROUTINE
00011'000000 FKEY:   0               ;FILTER FOR THE KEYBOARD
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
^ 0003  PE7.1
                ;       PROGRAMMING EXAMPLE 7.1
                ;
                ;THE FOLLOWING IS THE INTERRUPT HANDLER FOR THE APPRORIATE
                ;CLOCK ROUTINE (EITHER HRC OR RTC)

00012'054422 KEYINT:  STA 3,DISMIS      ;SAVE RETURN
00013'064436          DIA 1,GPR         ;INPUT KEYBOARD LINES
00014'125015          MOV# 1,1,SNR      ;SKP IF ANY KEY STRUCK
00015'000412          JMP NKEY          ;NO KEY WAS STRUCK

00016'010773          ISZ FKEY          ;INC FILTER
00017'002415          JMP @DISMIS       ;KEY NOT DOWN LONG ENOUGH
00020'024413          LDA 1,C60         ;THERE IS A KEY
                                        ;MAKE IT ASCII
00021'125223 KLOOP:   MOVZR 1,1,SNC     ;DECODE THE KEY
00022'000403          JMP DKEY          ;DONE
00023'101400          INC 0,0           ;KEY IN ASCII
00024'000775          JMP KLOOP

00025'006405 DKEY:    JSR @.PTKEY       ;GIVE KEY TO USER
00026'002406          JMP @DISMISS      ;DISMISS INTERRUPT

00027'024760 NKEY:    LDA 1,M5          ;NO KEY, RESET FILTER
00030'044761          STA 1,FKEY
00031'002403          JMP @DISMISS

00032'177777 .PTKEY:  PTKEY
00033'000060 C60:     60
00034'000000 DISMISS:         0

                      .END
```

ORIGINAL PAGE IS
OF POOR QUALITY

SECTION 8

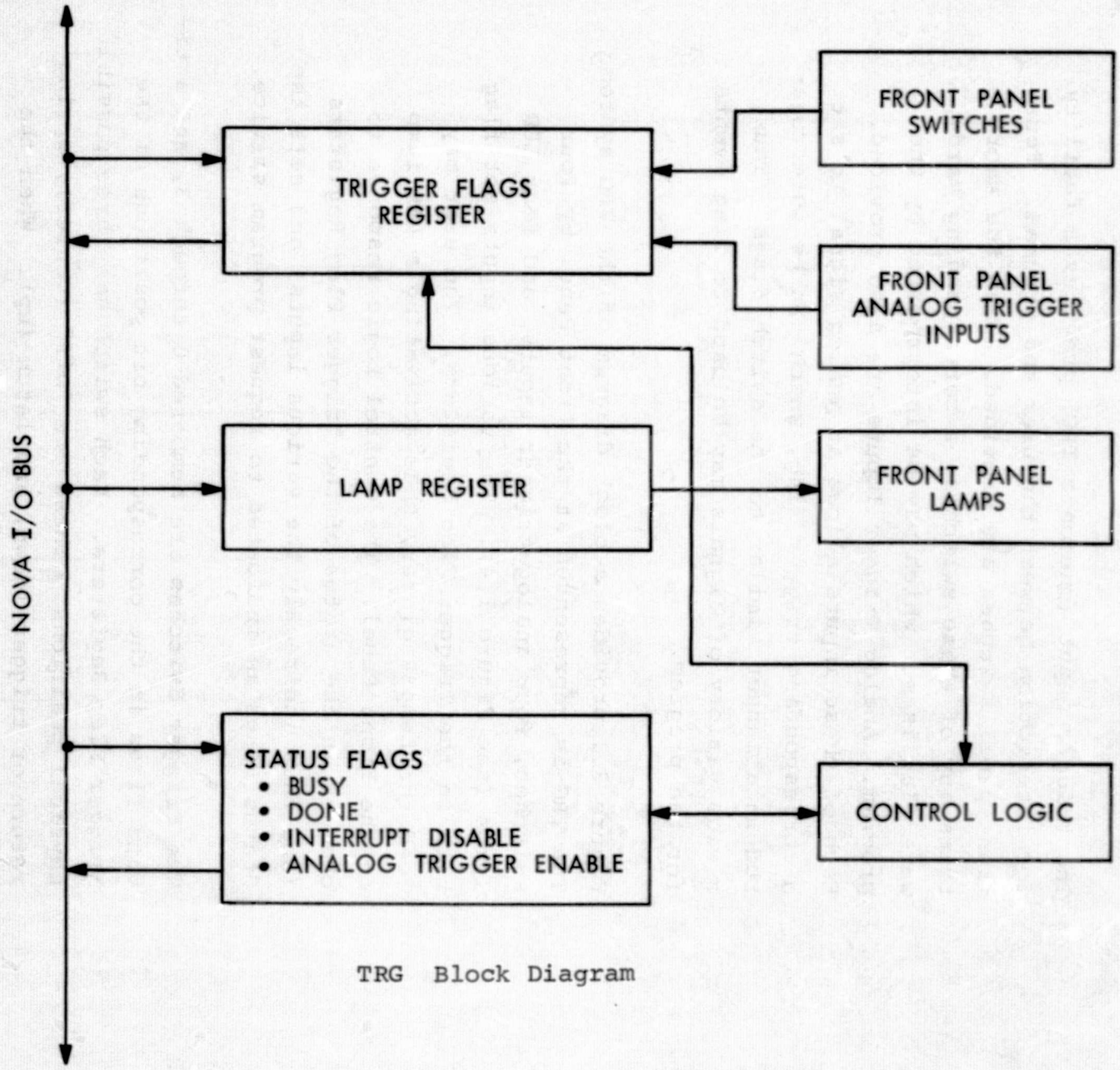
THE TRIGGER INPUTS

8.0 General Information

The trigger input (mnemonic TRG) provides a facility for interaction between the user and program. Four front panel switches are provided. When the user trips any of these switches, a corresponding hardware flag is set which may be interrogated by the program. Analog Trigger Inputs are also provided. Each of these inputs allows an analog signal to set a corresponding trigger flag, which can be interrogated in a manner similar to the switch flags. Thus, a wide variety of signals may be used to flag events for the program.

Figure 8.1 provides a block diagram of the TRG system. The TRG is represented at the Front Panel by four switches, four analog trigger inputs, and four LED lamps (see Figure 1.4). The various inputs set flag bits in the Trigger Flag Registers. The user may load the Lamp Register, thus activating a LED lamp on the Front Panel. The Control Logic responds to changes in the status of the Trigger Flag Registers (and thus indirectly the various inputs) and sets the Status Flags as indicated to request program service.

The Trigger Switches are numbered 0 through 3, and activate flags in the corresponding bit positions of the Trigger Flag Registers. Each switch has three possible physical positions. These are "off" (center), spring return or trigger (down) and latch (up). When the



TRG Block Diagram

switch is placed in the latch position, the corresponding Latch Flag is set. This flag may not be cleared by the program, thus it indicates the current physical position of the switch. If the switch is placed in the trigger position, it will return as soon as the user releases it. When the switch enters the trigger position, the corresponding Trigger Flag is set. This flag will remain set until cleared by the program, and indicates that the switch has entered the trigger position at some time in the past. Thus, two bits are used to describe the state of each switch. The latch bit is set if the switch is currently in the latch position. The trigger flag is set if the switch has been placed in the momentary position since the last time the flag was cleared.

Four analog trigger inputs are provided. Each of these inputs accepts analog inputs in the range $\pm$ 10 volts and compares the input with a reference value determined by the threshold knob (above and to the right of the input, see Figure 1.4). When the input signal exceeds the reference, an event is said to have occurred on that input. The user may consider the event to be equivalent to actuating a switch. Thus a trigger flag is set on the occurrance of an event, but will remain set even if the event goes away. Occurrance of another event while the corresponding Trigger Flag is set will result in the flag remaining set.

8.1 Analog Signal Input Specifications

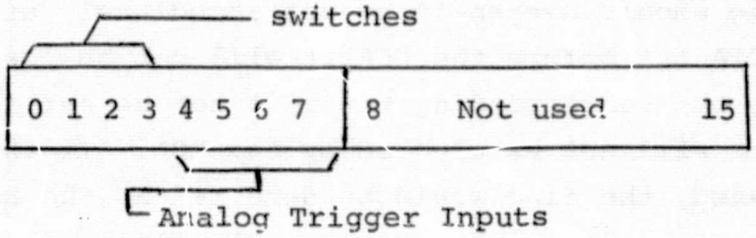
The Analog Trigger Inputs accept analog signals in the range $\pm$ 10 volts. The input has an impedance of 10 M ohms provided the input signal is in the normal operating range ($\pm$ 12 volts). If the signal exceeds the normal operating range, protective circuitry is activated to prevent damage to the electronics. If the protective circuits are activated, the input impedance will drop to 3,000 ohms. The device will recover from signals in the range $\pm$ 30 volts; if the overload signal exceeds this, a fusable element in the protective circuit will open thus disconnecting the offending signal from the electronics. If this occurs, the fusable element must be replaced before that input will function again.

As stated earlier, the analog signal is compared with a reference voltage determined by the setting of the threshold knob. The comparator is of the Schmitt type, with approximately 0.25 volts hystersis. This hystersis provides noise immunity, but requires that the peak-to-peak swing of the signal be greater than 0.25 volts for the circuit to trigger. The comparator operates up to 250 KHz thus easily exceeding any signal repetition rate that a program could respond to.

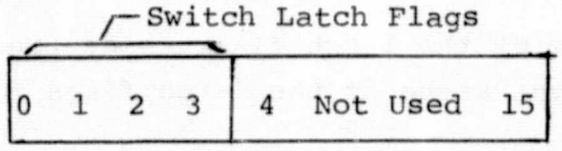
The input signal may be coupled into the comparator via direct or capacitive coupling. The capacitive coupling is convenient for removing large DC offsets on a small peak-to-peak signal. The user may also ground the input of the comparator, thus stopping any subsequent trigger action on that channel.

8.2 Trigger Flag Registers

Two registers in the TRG provide the user with the status bits for all of the device flags. The Trigger Flag Register (read in by a DIA d,TRG) provides the eight Trigger Flags--four for the switches and four for the analog trigger inputs. The Latch Register (read in with a DIB d,TRG) provides the four latch flags for the switches. The trigger flags are allocated as follows:



where the bit position in the register corresponds to the switch or analog trigger input number. The latch Flags are allocated as follows:



As before, the bit position corresponds to the switch number.

Once a flag has been set, the user must acknowledge receipt of the flag by "echoing" the flags he read. If the user echoes the state he has read in, it allows the TRG to compare the true state of the flags with with the state at the last time the user read the flags. In this manner, the TRG logic can determine when it is necessary to request program service to inform the program of a change in the status of the various flags.

The user may echo the state of the Trigger Flags by executing a DOA s,TRG; where the contents of the s register correspond to the states obtained by the user. Typically, the user would execute the sequence,

```
DIA 1,TRG          ;READ TRG FLAGS INTO BITS 0-7
DOA 1,TRG          ;ECHO FLAGS TO CLEAR THEM
```

This reads the flag status into AC1 then clears the corresponding flags in the device. Note that if a flag should happen to be set immediately after the DIA but before the DOA it will not be lost; since the corresponding bit will not be set in AC1 - and it will not be cleared by the DOA. If this happened, the flag would be detected at the next DIA instruction. The DOA actually clears the flags, thus a second DIA after the DOA in the above sequence would not have the flag bits set unless they were set between the DIA and DOA.

In the case of the latch flags, executing a DOB s,TRG will echo the state of the latch flags for the switches.

The user would execute;

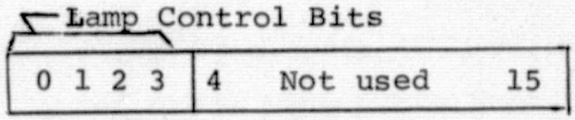
```
DIB 1,TRG          ;LATCH BITS INTO BITS 0-3
DOB 1,TRG          ;ECHO LATCH FLAGS
```

The Latch Flags are read into AC1, then echoed to the TRG. The DOB does not clear the latch flag, thus a DIB may be executed at any time to read the latch flags. The TRG compares the status of the flags as presented by the DOB instruction with the true physical position of the switches - then requests program service if they are different.

Both an IORST and a C pulse to the TRG will clear all flags.

8.3 Lamp Register

The user may load the lamp register with a DOC s,TRG instruction. The lamp control bits are assigned as follows:



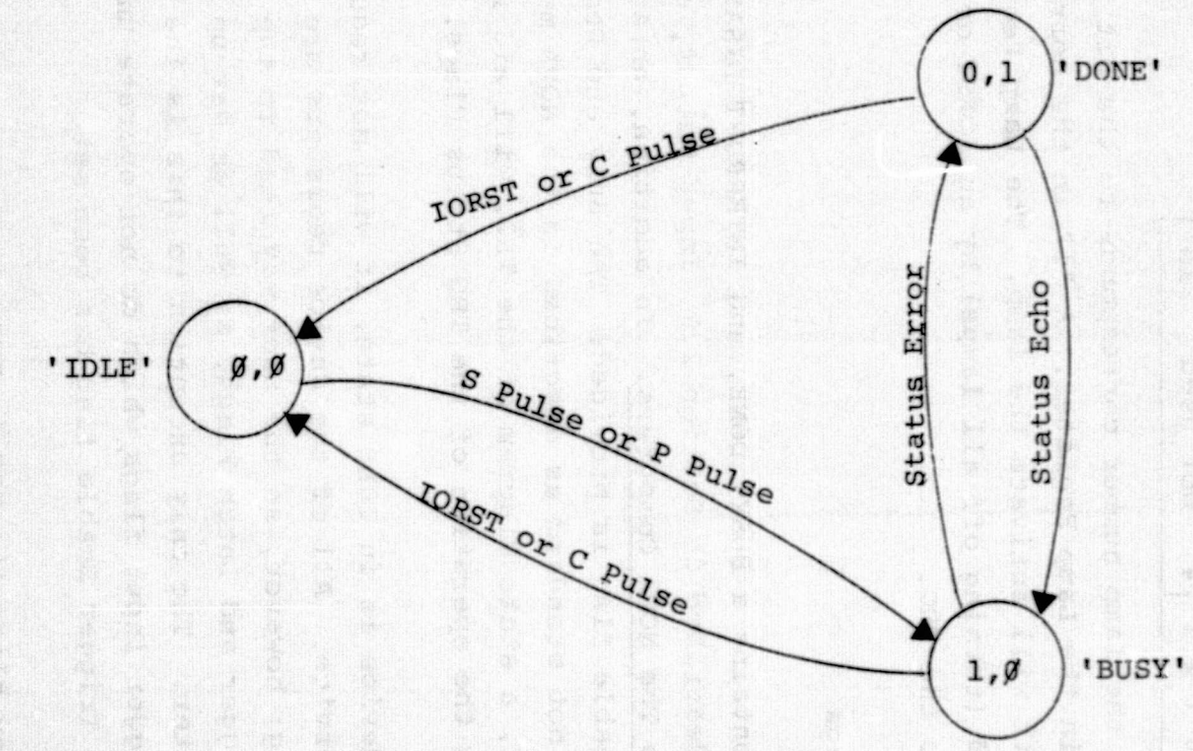
Note that the lamp number corresponds to the bit position in the Lamp Register. A "1" in the corresponding bit will activate the lamp. The Lamp Register is cleared (turning off all lamps) by an IORST or a C pulse to the TRG.

8.4 Status Flags

The TRG contains a BUSY, DONE, and INTERRUPT DISABLE flags as described in Section 2.3, Input-Output, of How to Use The NOVA Computers. In addition, an Analog Trigger Enable flag is provided. The Busy and Done flags are not standard as described in the NOVA manual. Figure 8.2, a state diagram of the TRG, will aid in explaining the operation of the TRG Status Flags.

When the device is in IDLE state, it will not request program service. All of the device registers are functioning; however, so the user may read in the state of the Trigger and Latch Flags, as well as set up the Lamp Register. The only exception to this is the analog trigger input flags, which do not operate unless the Analog Trigger Enable flag has been set.

The user may place the TRG in BUSY state by directing either an S pulse or P pulse to the device. If an S pulse is used, the Analog Enable Flag is set and both



KEY:

X-Y

X - Busy Flag State

Y - Done Flag State

m → -Transition initiated by m

FIGURE 8.2 - TRG State Transitions

Trigger Switches and Trigger Inputs are enable for action. If a P pulse is used, the Analog Trigger Enable Flag is cleared and only the Trigger Switches are enabled for action. This feature allows the program to handle the switches without having to process trigger inputs for the analog signals, which may occur very often.

If the TRG is in BUSY state and any of the Trigger or Latch Flags change state (differ from the last state update echoed by the program), then the TRG will enter DONE state. Once TRG is in DONE the user must echo a new and correct state update for the device to return to BUSY state.

An IORST or C pulse will always return the device to IDLE state, as well as clear the Trigger, Latch, and Lamp Registers.

The Interrupt Disable Flag (connected to bit 6, mnemonic MSTRG) operates in all respects as described in Section 2 of How to Use the NOVA Computers. That is, if the device is in DONE state, and Interrupt Disable is cleared, the device will request interrupt service.

If the device is in DONE state, it will respond to the INTA instruction, returning the device code, "TRG".

8.5 Programming the TRG

In the simplest case, the user might want to simply determine the status of the switches occasionally in his program. To do this, he need only execute the following sequence,

```
DIA 1,TRG      ;READ TRIGGER FLAGS
DOA 1,TRG      ;AND CLEAR THEM

DIB 2,TRG      ;READ LATCH FLAGS
```

This code will return the status of the Trigger Flags in AC1, and the status of the Latch Flags in AC2. The 'DOA 1,TRG clears the Trigger Flags every time we read them, so that the contents of AC1 tell the user which flags have been set since the last time he read them in. The user need not explicitly start the TRG unless he must use the Analog Trigger Inputs. In that case, an S pulse must be used to start the TRG so that the Analog Trigger Enable Flag will be set.

If in the above example the user wished to provide the operator with visual feedback of the status of the Trigger Flags, he could simply add a "DOC 1,TRG to the sequence. This would light the front panel lamp corresponding to every Trigger Flag that was set.

Programming Example 8.1 details a routine which reads in the TRG Flags, then encodes unprocessed flags on a priority basis.

```
0001  PE8.1
;       (56,1599)          JBW       MAY 2,1973          PAGE 1
;
;       PROGRAMMING EXAMPLE 8.1
;       ENCODING THE TRIGGER/LATCH INPUTS
;
;       ABSTRACT:
;
;       '.TRGEC' READS IN THE CONTENTS OF BOTH
;       THE TRIGGER AND LATCH FLAG REGISTERS.
;       EACH REGISTER IS ENCODED AS A BINARY NUMBER
;       REPRESENTING THE BIT POSITION OF THE HIGHEST
;       PRIORITY FLAG SET.  THE PRIORITY DECENDS FROM
;       LEFT TO RIGHT IN THE WORD.
;
;       CALLING SEQUENCE:
;
;       JSR @=.TRGEC        ;BRANCH TO '.TRGEC' USING A
;                           ; LOCAL ADDRESS CONSTANT.
;       <RETURN>            ;RETURN HERE WITH SWITCHES
;                           ; ENCODED.
;
;       THE USER MUST INITIALIZE THE TRG WITH AN
;       IORST OR C PULSE TO CLEAR ANY FLAGS SET AT
;       POWER TURN ON TIME.  '.TRGEC' DOES NOT SKIP ON
;       THE TRG BUSY/DONE FLAGS;  BUT ALWAYS ECHOS
;       THE LATEST STATUS, SO THAT THE USER MAY
;       SKIP ON THE BUSY/DONE FLAGS.
;
;       ARGUMENTS:  NONE.
;
;       RETURNS:
;
;       WHEN THE RETURN IS TAKEN, AC0 CONTAINS THE
;       ENCODED TRIGGER FLAG AND AC1 CONTAINS THE
;       ENCODED LATCH FLAG.  THE ENCODED TRIGGER FLAG
;       IS IN THE RANGE 0 TO 7, WHILE THE LATCH FLAG
;       WILL BE IN THE RANGE 0 TO 3.  IF THE REGISTER
;       IS ALL ONES, THEN NO FLAG WAS SET.
;       THE UPDATED STATUS IS ECHOED TO THE TRG,
;       CLEARING ALL FLAGS SET AT THAT TIME.  THE
;       ROUTINE MAINTAINS A CORE COPY OF THE
;       STATUS SO THAT ALL FLAGS WILL BE PROCESSED
;       BY THE SOFTWARE.
;       '.TRGEC' SHOULD BE REPEATEDLY CALLED, UNTIL
;       ALL ONES IS RETURNED, INDICATING
;       NO MORE FLAGS TO BE PROCESSED.
;
;       ACCUMULATORS:
;
;       ALL AC'S AND CARRY ARE DESTROYED.
;
;       SPECIAL CONSIDERATIONS:
;
;       .TRGEC USES THE '.OR' ROUTINE FROM THE
;       DCG MATH.LB FILE.  THUS THE MATH LIBRARY
;       MUST BE INCLUDED IN THE LOADER COMMAND.
```

ORIGINAL PAGE IS OF POOR QUALITY

```
^ 0002  PE8.1
                ;       (45,981)        JBW     MAY 2,1973      PAGE 2
                ;
                ;       PROGRAMMING EXAMPLE 8.1 - CONT.
                ;
                ;       ENCODING TRIGGER/LATCH FLAGS.
                ;
                        .TITLE PE8.1
                        .ENT .TRGEC
                        .EXTN .OR       ;LOGICAL OR FUNCTION IN MATH.LB
                        .NREL

00000' 054444 .TRGEC:   STA 3,SV3.0     ;SAVE RETURN - NOTHING ELSE

00001' 060437           DIA 0,TRG       ;GET THE TRIGGER FLAGS
00002' 061037           DOA 0,TRG       ; CLEAR THE FLAGS IN THE DEVICE

00003' 024437           LDA 1,OLDTF     ;GET OLD TRIGGER FLAGS

00004' 006435           JSR @OR         ;OR THE NEW AND OLD TO GET ALL
                                        ; UNPROCESSED IN AC0
00005' 030433           LDA 2,CN8.      ;GET # OF TRG FLAGS

00006' 004412           JSR PRIOR       ;ENCODE BY PRIORTY

00007' 040433           STA 0,OLDTF     ;SAVE OLD TRG FLAGS - I.E. ALL
                                        ; EXCEPT THE ONE JUST ENCODED
00010' 054433           STA 3,TRNUM     ;SAVE ENCODED FLAG #

00011' 061437           DIB 0,TRG       ;GET LATCH FLAGS
00012' 062037           DOB 0,TRG       ;ECHO THEM - DOES NOT CLEAR THEM
                                        ; BUT LETS USER USE BUSY,DONE

                ;       >> THERE ARE NO OLD LATCH FLAGS, SINCE THESE
                ;          FLAGS CORRESPOND TO TRUE PHYSICAL SWITCH
                ;          POSITION.

00013' 030424           LDA 2,CN4       ;ONLY 4 LATCH BITS

00014' 004404           JSR PRIOR       ;ENCODE BY PRIORITY

00015' 165000           MOV 3,1         ;RETURN ENCODED LATCH IN AC1
00016' 020425           LDA 0,TRNUM     ;RETURN ENCODED TRIGGER IN AC0

00017' 002425           JMP @SV3.0      ;RETURN TO CALLER
```

```
^ 0003  PE8.1
                ;       (57,1308)       JBW     MAY 2,1973      PAGE 3
                ;
                ;       PROGRAMMING EXAMPLE 8.1

                ;       ENCODING TRIGGER/LATCH FLAGS


                ;       ROUTINE TO ENCODE AC0.

                ;       AC0 IS SHIFTED TO THE LEFT, UNTIL THE
                ;       COUNT PASSED IN AC2 IS EXHAUSTED OR A BIT
                ;       SET TO '1' IS FOUND.  IF THE COUNT
                ;       IS EXHAUSTED, AC3 WILL CONTAIN ALL ONES
                ;       ON RETURN.  IF A BIT IS FOUND SET, AC3 WILL
                ;       CONTAIN THE # OF ITS BIT POSITION.  THE
                ;       BIT FOUND IS CLEARED FROM AC0, WHICH IS
                ;       RETURNED WITH ALL OTHER BITS UNMODIFIED.

00020' 054425 PRIOR:    STA 3,SV3.1         ;SAVE RETURN ADDRESS
00021' 040425           STA 0,SV0.1         ;SAVE AC0 FOR RETURN

00022' 176400           SUB 3,3             ;START AT BIT POSITION ZERO
00023' 126220           ADCZR   1,1         ;MAKE A MASK WITH BIT0 = 0
                                            ;ALL OTHERS = 1
00024' 101122 LP:       MOVZL 0,0,SZC       ;SKP IF '0' FOUND
00025' 000407           JMP ONEFND          ;JMP IF A '1' THERE

00026' 175400           INC 3,3             ;COUNT OFF ONE MORE BIT POSITION
00027' 125220           MOVZR 1,1           ;SHIFT MASK OVER, ADDING ZEROS

00030' 151404           INC 2,2,SZR         ;SKP IF TESTED ALL BITS
00031' 000773           JMP LP              ;NO - LOOK AT NEXT BIT POSITION

00032' 176000           ADC 3,3             ;NOTHING FOUND - FLAG IT
00033' 002412           JMP @SV3.1          ;RETURN

00034' 020412 ONEFND:   LDA 0,SV0.1         ;GET ORIGINAL BIT PATTERN
00035' 123400           AND 1,0             ;MASK OUT BIT WE FOUND,OTHERS
                                            ;LEFT ALONE
00036' 002407           JMP @SV3.1          ;RETURN


                ;       CONSTANTS AND WORKING SPACE

00037' 177774 CN4:      -4          ;FOUR LATCH BITS
00040' 177770 CN8.:     -8.         ;EIGHT TRIGGER FLAGS
00041' 177777 OR:       .OR         ;LOGICAL OR ROUTINE

00042' 000000 OLDTF:    0           ;THE LIST OF TRG FLAGS NOT PROCESSED YET
       000001 TRNUM:    .BLK 1      ;TEMP FOR ENCODED TRG #

       000001 SV3.0:    .BLK 1      ;SAVE AREA
       000001 SV3.1:    .BLK 1
       000001 SV0.1:    .BLK 1

                        .END
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX A

SPECIAL ASSEMBLY LANGUAGE DEFINITIONS

PRECEDING PAGE BLANK NOT FILMED

```
001  .MAIN
                ;       (52,1556) JBW   11/26/71
                ;                 JBW   12/15/72

                ;       CBL NOVA ASSEMBLY DEFINITIONS

                ;       THIS TAPE DEFINES THE SPECIAL SYMBOLS REQUIRED
                ;       TO REFERENCE THE ANALOG INTERFACE AS WELL AS
                ;       A FEW CONVIENT INSTRUCTION CODES NOT PROVIDED
                ;       IN THE STANDARD DGC ASSEMBLER.


                ;       ANALOG INTERFACE DEVICE DEFINITIONS

                ;               NOTE THAT BY CHANGING ONLY THE VALUE
                ;               OF 'AIN', ALL OTHER DEVICES ARE
                ;               MAPPED TO THE CORRECT ADDRESS RANGE.

000031 .DUSR    AIN =31             ;ANALOG INPUT
000032 .DUSR    HRC =AIN+1          ;HIGH RESOLUTION CLOCK
000033 .DUSR    AOUT=HRC+1          ;DUAL CHANNEL ANALOG OUTPUT
000034 .DUSR    DISP=AOUT+1         ;X-Y DISPLAY
000035 .DUSR    CHAR=DISP+1         ;CHARACTER GENERATOR
000036 .DUSR    GPR =CHAR+1         ;GENERAL PURPOSE REGISTER
000037 .DUSR    TRG =GPR+1          ;TRIGGER INPUTS

000074 .DUSR    LINC=74             ;LINC TAPE DRIVE

                ;       MASK DEFINITIONS FOR ALL DEVICES

100000 .DUSR    MSDCM =1B0          ;DATA COMMUNICATIONS MULTIPLEXER
100000 .DUSR    MSAIN =1B0          ;ANALOG INPUT
040000 .DUSR    MSHRC =1B1          ;HIGH RESOLUTION CLOCK
020000 .DUSR    MSAOUT=1B2          ;ANALOG OUTPUT
010000 .DUSR    MSDISP=1B3          ;X-Y DISPLAY
010000 .DUSR    MSCHAR=1B3          ;CHAR GENERATOR
004000 .DUSR    MSLINC=1B4          ;LINC TAPE
002000 .DUSR    MSGPR =1B5          ;GENERAL PURPOSE REGISTER
001000 .DUSR    MSTRG =1B6          ;TRIGGER INPUTS
000400 .DUSR    MSDKP =1B7          ;MOVING HEAD DISK
000200 .DUSR    MSADCV=1B8          ;DCG A/D CONVERTER
000100 .DUSR    MSDSK =1B9          ;DISK
000040 .DUSR    MSMTA =1B10         ;MAG TAPE
000040 .DUSR    MSCDR =1B10         ;CARD READER
000020 .DUSR    MSPTR =1B11         ;PAPER TAPE READER
000010 .DUSR    MSMCAT=1B12         ;MULTIPROCESSOR TRANSMITTER
000010 .DUSR    MSMCAR=1B12         ;MULTIPROCESSOR RECEIVER
000010 .DUSR    MSPLT =1B12         ;INCREMENTAL PLOTTER
000010 .DUSR    MSLPT =1B12         ;LINE PRINTER
000004 .DUSR    MSPTP =1B13         ;PAPER TAPE PUNCH
000004 .DUSR    MSRTC =1B13         ;REAL TIME CLOCK
000002 .DUSR    MSTTI =1B14         ;TELETYPE INPUT
000002 .DUSR    MSAHM =1B14         ;ASYNC. HARD. MULTI.
```

```
0002  .MAIN
                ;       (57,1562) JBW   11/26/71

        000002          .RDX 2                  ;ALL ARE BIT PATTERNS

                ;       AIN CONTROL REGISTER BITS

        000020 .DUSR    AINSCAN =1B11   ;SETS AIN IN AUTO SCAN MODE
                                        ; A '0' IN BIT 11 SELECTS
                                        ; NON SCAN MODE.
        000000 .DUSR    AINSP   =00     ;SELECTS S PULSE CONVERT COMMAND
        000001 .DUSR    AINHRC  =01     ;SELECTS HRC OVERFLOW
        000003 .DUSR    AINEXT  =11     ;SELECTS EXTERNAL CONTROL PULSE

                ;       HRC CONTROL REGISTER BITS

        000020 .DUSR    HRCOVFL =1B11   ;ENABLES OVERFLOW TO SET DONE
                                        ; A '0' SUPPRESSES OVERFLOW FLAG
        000010 .DUSR    HRCEND  =1B12   ;ENABLES EXT. END. TO SET DONE
                                        ; A '0' SUPPRESSES END FLAG
        000004 .DUSR    HRCEXT  =1B13   ;SELECTS EXT. CLOCK
                                        ; A '0' SELLECTS THE INTERNAL
                                        ; 1MHZ. CRYSTAL CLOCK

                ;       AOUT CONTROL REGISTER BITS

        000000 .DUSR    AOUTSP  =00     ;SELECTS S PULSE AS CONVERT COM.
        000001 .DUSR    AOUTHRC =01     ;SELECTS HRC
        000002 .DUSR    AOUTTRK =10     ;SELECTS TRACK MODE
        000003 .DUSR    AOUTEXT =11     ;SELECTS EXT. CONTROL LINE

                ;       DISP CONTROL REGISTER BITS

        000000 .DUSR    DISPSP  =00     ;SELECTS S PULSE TO INTEN BEAM
        000001 .DUSR    DISPHRC =01     ;SELECTS HRC OVERFLOW
        000002 .DUSR    DISPTRK =10     ;SELECTS TRACK MODE
        00000? .DUSR    DISPEXT =11     ;SELECTS EXT. CONTROL LINE

        004000 .DUSR    DISP0   =1B4    ;SELECTS X-Y CHAN 0, BY
                                        ; CONVENTION THE SMALLEST XY
                                        ; SCOPE ON THE SYSTEM.
        002000 .DUSR    DISP1   =1B5    ;CHAN1, BY CONVENTION THE
                                        ; LARGER SCOPE, IF A TWO
                                        ; SCOPE SYSTEM.
        001000 .DUSR    DISP2   =1B6    ;CHAN2, BY CONVENTION THE
                                        ; HARDCOPY XY PLOTTER.
        000400 .DUSR    DISP3   =1B7    ;CHAN3

        000040 .DUSR    PENDWN  =1B10   ;PUTS PEN DOWN ON PLOTTER

                ;       CHARACTER GENERATOR SIZE CONTROL

        000000 .DUSR    CHARS   =00B7   ;SMALL CHARACTERS - DOT SPACING
                                        ; OF 2.
        000400 .DUSR    CHARM   =01B7   ;MEDIUM CHARACTERS - DOT SPACING
                                        ; OF 3.
        001000 .DUSR    CHARL   =10B7   ;LARGE CHARACTERS - DOT SPACING
                                        ; OF 6.
```

```
^ 0003  .MAIN
                ;       (52,1303) JBW   11/26/71
                ;                       4/27/73
                ;       SPECIAL INSTRUCTION DEFINITIONS

        000010          .RDX 8                  ;BACK TO NORMAL

                ;       SINGLE INSTRUCTION SIGNED ARITHMETIC
                ;       COMPARISONS

                ;       -----WARNING-----
                ;       THESE SIGNED TESTS ONLY WORK IF THE
                ;       SIGNED RESULT IMPLIED BY THE DEFINITION
                ;       CAN BE REPRESENTED IN 16 BITS.  FOR
                ;       EXAMPLE, IN THE TEST 'SLE' A SUBTRACTION
                ;       IS REQUIRED.  IF THE SOURCE WERE -32767
                ;       (100001 OCTAL) AND THE DESTINATION WERE
                ;       1 (000001), THEN THE RESULT OF A 'SUB S,D'
                ;       WOULD BE -32768 (100000) DUE TO LOSS OF
                ;       THE SIGN BIT DURING THE OPERATION.
                ;       IN THIS CASE THE SKIP WOULD FAIL.

                ;       THE GENERAL RULE IS THAT THE TRUE
                ;       RESULT OF THE OPERATION USED IN THE
                ;       TEST MUST FALL IN THE RANGE -32768 TO
                ;       32767.

        102512  .DALC   SLE=SUBL# 0,0,SZC       ;SKIP IF S <= D
        102513  .DALC   SGT=SUBL# 0,0,SNC       ;SKIP IF S > D
        102414  .DALC   SEQ=SUB#  0,0,SZR       ;SKIP IF S = D
        102415  .DALC   SNE=SUB#  0,0,SNR       ;SKIP IF S NOT= D
        102112  .DALC   SLT=ADCL# 0,0,SZC       ;SKIP IF S < D
        102113  .DALC   SGE=ADCL# 0,0,SNC       ;SKIP IF S >= D


                ;       SINGLE INSTRUCTION UNSIGNED ARITHMETIC
                ;       COMPARISONS

                ;       THESE COMPARISONS WORK FOR ALL 16 BIT
                ;       UNSIGNED NUMBERS.

        102433  .DALC   USLE=SUBZ# 0,0,SNC       ;SKIP IF S <= D
        102432  .DALC   USGT=SUBZ# 0,0,SZC       ;SKIP IF S > D
        102033  .DALC   USLT=ADCZ# 0,0,SNC       ;SKIP IF S < D
        102032  .DALC   USGE=ADCZ# 0,0,SZC       ;SKIP IF S >= D


                ;       A NO OPERATION INSTRUCTION

        100010  .DUSR   NOP=COM# 0,0    ;A NULL ARITH.  INST.

                .END
```

APPENDIX B

GENERAL PHYSICAL AND ELECTRICAL SPECIFICATIONS

Characteristic	Specification	Comments
Physical Specifications:		
Dimensions		
Height	5-1/4 inches	Unit is designed to rack mount in a Tektronics 5-1/4-inch rack mounting adaptor for Type 601 or 602 Displays. Tektronics Part Number 016-0115-02.
Width	8-1/2 inches	
Depth	17 inches	
Electrical Specifications:		
Line Voltage	105 to 130 VAC	Line voltage and frequency specifications are those quoted by power supply manufacturer.
Line Frequency	50 to 1000 Hz.	
Fusing	2A, Fast Blow	Fuse type AGC 2.
Line Power Control		
ON	Power On	Set by switch on rear connector panel. Some units may not have the AUTO power logic installed. For these units, it is recommended that the unit be plugged into the NOVA switched line outlet, and left in the ON position.
OFF	Power Off	
AUTO	Power controlled by NOVA I/O Bus power flag.	
Switched Outlet	1/4 Amp maximum at line voltage and frequency.	Line voltage appears here whenever the interface has internal power. It is recommended that the cooling fan be powered by this outlet.

Mounting and Ventilation:

For proper cooling, a 50 to 100 cubic feet per minute fan must be provided to ventilate the unit. The fan should be mounted so as to displace air past the power supply and/or through the cards. In order to facilitate ventilation, free airspace must be provided above and/or below the unit.

APPENDIX C

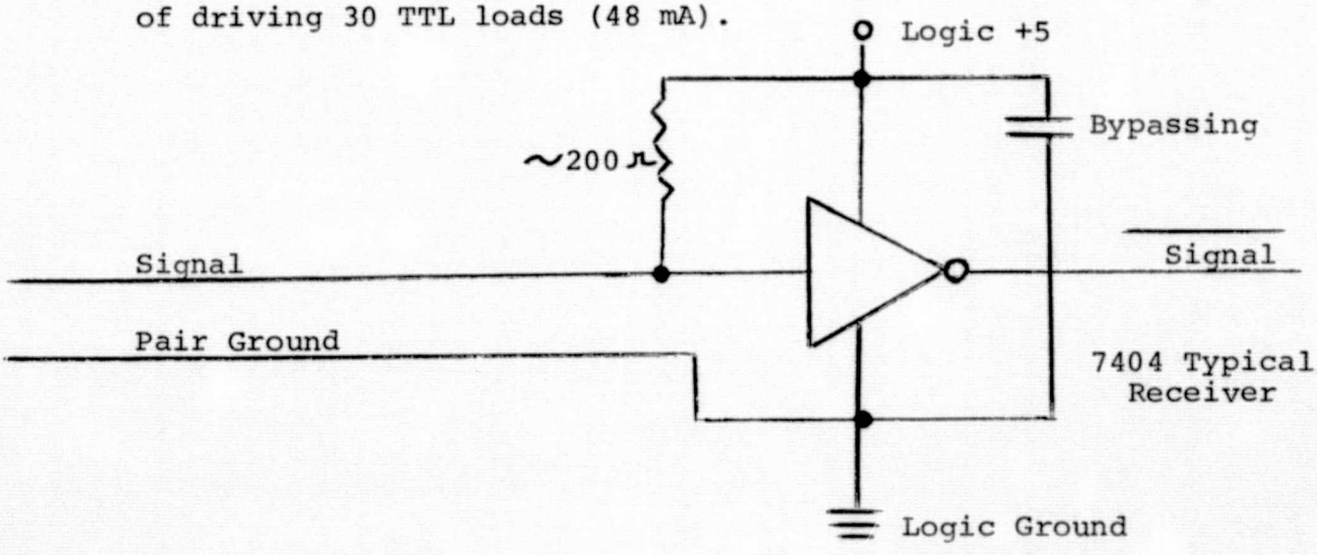
INTERFACE SIGNAL SPECIFICATIONS AND CONNECTOR ALLOCATION

C.1 Driving and Receiving External Lines

All logic levels in the Analog Interface are standard TTL levels; the standard load is one Texas Instruments 7400 series load of 1.6mA.

In order for the user to interface special devices to the Analog Interface, he must make up cables for the various logic signals involved. Since all the signal lines will, of necessity, be several feet in length, proper consideration must be given to termination of the signal lines. All signal lines should be twisted pairs, with the ground side of the pair brought to ground at both ends.

If the user is receiving a signal line from the Analog Interface, he must terminate the line to assure proper operation. The following termination scheme is recommended for signal lines specified as capable of driving 30 TTL loads (48 mA).

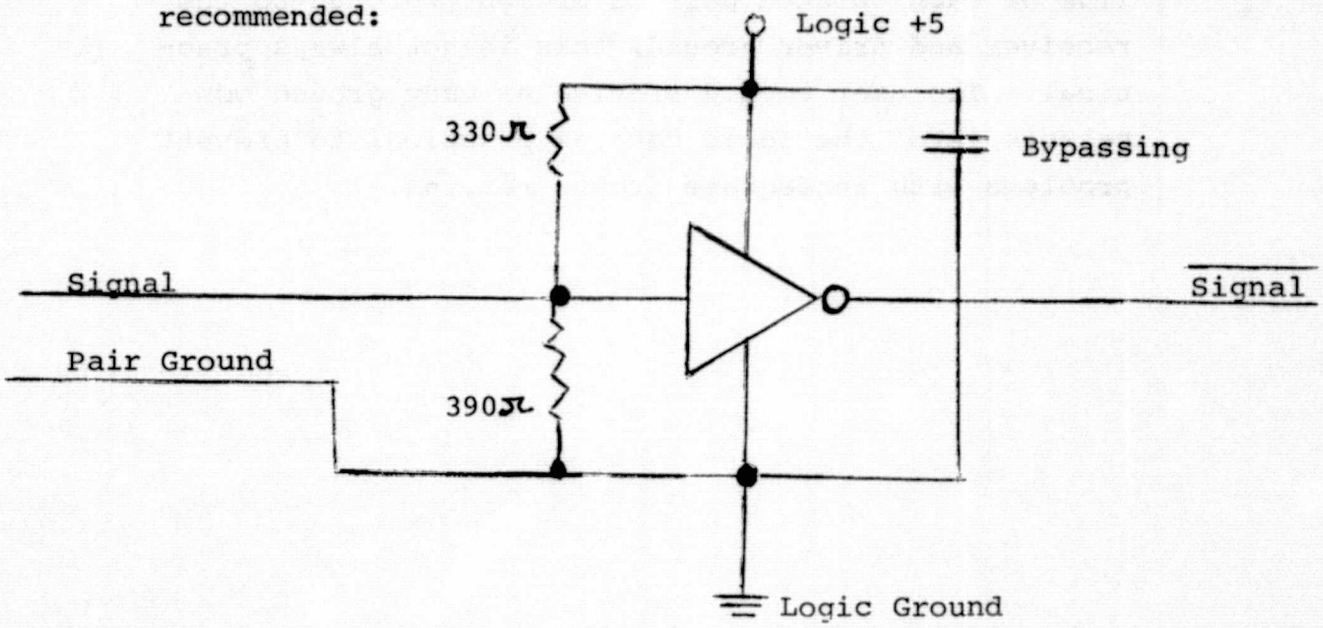


This method provides a termination for the line and good noise margins. Particular care should be taken in bypassing the receiver and termination resistors. At least 0.01μF

should be alloted for every four outputs at the receiver, while the termination resistors should have at least 0.01μF per eight lines. The capacitors must be RF quality such as disk ceramics, and mounted as close to the device being bypassed as possible.

While it is acceptable to use a two input gate as a receiver by tying both inputs to the signal line, it is not recommended that any gating functions be preformed on the signal by the receiver.

For signal lines specified as capable of driving 10 standard loads, the following terminations scheme is recommended:



As for the previous termination, special attention should be placed on bypassing the receiver and termination resistors.

It is very important to carry the ground line of the pair as directly as possible to the receiver ground. The receiver will, of course, be grounded to the logic ground in the users' device.

If the user is driving lines into the Analog Interface, he need not terminate them, since the Analog Interface internally terminates input lines. From the signal specification charts, the user can determine the drive capability required of his circuitry by the Analog Interface. Recommended line drivers are 7437's, 7438's, 75451's, or 75452's. The ground line of the pair the user is driving should be brought to the ground of the driver package. All packages driving lines should have 0.1μF bypassing capacitors within 1-1/2 inches of the package.

While the above guidelines recommend that the ground line of each twisted pair be brought through to the receiver and driver ground, this is not always practical. The user should provide as many ground connectors into the logic card as practical to prevent problems with inadequate ground returns.

C.2 HRC Signal Specifications

Signal	Characteristics	Specifications	Comments
HRC EXTERNAL CLOCK*	Pulse width repetition rate loading.	≥ 0.25 μSec ≤ 1 MHz 30 TTL loads	This signal is selected as the basic reference clock by HRCEXT mode. This is an edge clocking signal, active on the high (+5 V) to low (Ø V) transition. Available at TRG connector and NOVA backpanel pin B51 on the Analog Interface slot.
HRC EXTERNAL END*	Pulse width loading	≥ 0.25 μSec. 30 TTL loads	This signal is enabled for control by HRCEND mode. This is an edgeclocking signal, active on the high (+5) to low (Ø V) transition. Available at TRG connector and NOVA backpanel pin B53 on the Analog Interface slot.
HRC OVERFLOW PULSE*	Pulse width drive	0.25 to 0.75 μsec. 30 TTL loads	This pulse is generated by every overflow of the Counting Register. Available at NOVA backpanel pin B49 on the Analog Interface slot.

C.3 AIN Signal Specifications

Signal	Characteristic	Specification	Comment
Front Panel Analog Inputs	Valid Input conversion range	0.4 volts p-p to 8.0 volts p-p	Continuous gain control (0.5 to 10) allows scaling of input signal to nominal ± 2 V channel multiplexer range.
	maximum continuous input level	± 30 volts	Inputs are protected against excessive input voltages by fusable resistors. If the input exceeds this range for a substantial period, the input line will be opened to protect the amplifier.
	Input DC Resistance signal in range ± 12 volts.	10 M ohms ± 10%	
	signal exceeds range ± 12 volts.	3 K ohms	Input protective circuit begins to affect input impedance at about ± 12 volts. Protective circuits are after coupling capacitors.
	Offset adjustment	± 5 volts	Offset is summed with input signal after coupling, but before gain control.
	Gandwidth		
	DC coupled		
	gain = 1	DC to ≥250 KHz @-3dB	
	gain = 10	DC to ≥25 KHz @-3dB	
	AC coupled		
	gain = 1	≤1 to ≥250 KHz @-3dB	
	gain = 10	≤1 to ≥25 KHz @-3dB	
	Risetime	≤2 μsec.	10% to 90% points, unity gain, 4 volt steph.

C.3 AIN Signal Specifications Continued.....

Signal	Characteristic	Specification	Comment
Rear Panel Analog Inputs	Valid Input conversion range	$\pm$ 2 volts	Precise range determined by gain adjustment on analog to digital converter. Nominal least significant bit is 4 mV.

All other Rear Panel Analog Input Specifications are the same as those of the Front Panel Analog Inputs, with the exceptions that there is no offset adjustment and bandwidth is specified only for DC coupling at unity gain.

AIN EXTERNAL STROBE*	Width Loading	$\geq$0.25μsec. 30 TTL loads	This signal is selected to initiate conversions by the AINEXT mode. This is an edge clocking signal, active on the high (+5 V) to low (Ø V) transition.

C.4 AOUT Signal Specifications

Signal	Characteristic	Specifications	Comments
A & B Analog Output	range	±2 Volts	Minimum load for specified output range is 1K ohms. The specifications are those of the LM310 followers used. Settling time will vary with cable capacitance on the output.
	output impedance	2.5 ohms	
	output current	±2 mA	
	settling time	~5μsec.	
AOUT EXTERNAL STROBE*	pulse width	≥0.25	This signal may be selected to initiate a conversion by AOUTEXT mode. The high (+5 v) to low (Ø V) transition on this line initiates the output conversion.
	loading	30 TTL loads	

C.5 DISP Signal Specifications

Signal	Characteristic	Specification	Comments
X & Y Analog Output	Range Output impedance Output current Settling time	±2 Volts 2.5 ohms ±2 mA ~5μsec.	Minimum load for specified output range is 1K ohms. The specifications are those of the LM310 followers used as output buffers. Settling time will vary with cable capacitance on the output. Every display channel has independent output buffers so these specifications apply to each output connector X-Y signals.
INTEN	pulse width drive	4 to 20μsec. 30 TTL loads	Adjustable to suit the standard highspeed display device in the system.
INTEN*			Complement of INTEN, same specification.
XY SELECT	drive	10 TTL loads	This line is activated by the corresponding bit in the DISP Control Register.
XY OPTION ENABLE	drive	10 TTL loads	This line is activated by the corresponding bit in the DISP Control Register.
XY EXTEND*	loading	30 TTL loads	Driven by user device interface to extend plotting cycle. Must be driven by an open collector gate. For the extend function to operate, the user must activate this line at least 2μsec. before the end of INTEN. The device may hold the line active as long as required. A new plot cycle may start within a few microseconds after the release of this line.

C.5 DISP Signal Sepcifications Continued.....

Signal	Characteristic	Specification	Comments
XY WAIT*	Loading	30 TTL loads	Driven by the user device interface to inhibit changes in the X-Y analog output signals. Must be driven by an open collector gate. In normal operation, the DISP logic will wait 5 μsec. after a change in the Control Register (SELECT & OPTION) before changing the X-Y analog outputs. If the user device activates XY WAIT within 3 μsec. after a change in XY SELECT or XY OPTION ENABLE, he may extend the normal 5 μsec. waiting period as required for the device. The user may also activate XY WAIT at any time to suppress changes in the X-Y analog outputs. Note that a change in X and Y could have started just before the user activated the XY WAIT line. In this case, the X and Y outputs would settle to the new position, then remain fixed until XY WAIT was released by the Interface. If activation of XY WAIT is not synchronized with changes in XY OPTION ENABLE, then it is possible for one INTEN Pulse to start within 2 μsec. after XY WAIT is activated.
DISP EXT STROBE*	Pulse width loading	≥ 0.25 μsec. 30 TTL loads	This signal may be selected to initiate a conversion by DISPEXT mode. The high (+5 V) to low (Ø V) transition on this line initiates the conversion.

C.6 GPR Signal Specifications

Signal	Characteristic	Specifications	Comment
USER COUNT*	Width Repetition rate Loading	$\geq$ 0.25 μsec. $\leq$ 2 M Hz 30 TTL loads	A pulse on this line causes the Input Register to increment on the high (+5 V) to low (Ø V) transition of this line.
USER STRB*	Width loading	$\geq$ 0.25 μsec. 30 TTL loads	A pulse on this line strobes the DATAIN Lines into the Input Register. The DATAIN lines must be at the correct value at the start of USER STRB, and must be held at the correct value for 0.25 μsec. after USER STRB ends. A USER COUNT pulse cannot occur for 0.25 μsec. after the end of USER STRB.
REGRST*	Width Loading	$\geq$ 0.25 μsec. 30 TTL loads	A pulse on this line resets the Input register to all zeros. A USER COUNT cannot occur for 0.1 μsec. after the end of REGRST..
USER DEV COM*	Width Loading	$\geq$ 0.25 μsec. 30 TTL loads	If GPR BUSY is set, a pulse on this line sets DONE and clears BUSY. The transition from high (+5 V) to low (0 V) on this line clocks the GPR status change.

C.6 GPR Signal Specifications Continued.....

Signal	Characteristic	Specifications	Comments
GPR,DOB GPR DOC, GPR DIC, GPR S PULSE, GPR C Pulse, and GPR P Pulse	Drive	30 TTL loads	All of these signals are derived from the corresponding NOVA Bus signals gated with GPR select on the NOVA I/O Bus. Their precise timing is depend-on the host NOVA processor, see How to Use the NOVA Computers for detailed timing.
GPR BUSY, GPR DONE	Drive	30 TTL loads	These signals reflect the Current Status of the GPR
DATAINØ-15*	Load	30 TTL loads	
DATAOUT-15*	Drive	30 TTL Loads	

C.7 TRG Signal Specifications

Signal	Characteristic	Specifications	Comments
Front Panel Analog Trigger Inputs	Input Usable Range	20 volts p-p	This assumes signal swing is sufficient to drop below the trigger level setting. If DC coupled, this range is centered around zero.
	Maximum Continuous Input level	± 30 volts	Inputs are protected against excessive voltages by fusable elements. If the input exceed this range, the input line may be opened.
	Trigger Level Range	± 5 volts	Continuously adjustable
	Hysteresis Level	~0.25 volts	p-p signal level must exceed this level for proper triggering.
	Trigger Rate	≥250 K Hz.	1 volt p-p square wave.
	Input DC Resistance Signal in range ± 12 volts.	10 M ohms ±10%	Input protective circuit begins to affect input impedance at about ± 12 volts. Protective circuits are after coupling capacitors.
	Signal exceeds range of ± 12 volts.	≥3 K ohms	
BUFFERED TRIGGER 4-7*	Drive capability	30 TTL Loads	These signals are the buffered outputs of the Analog Trigger Inputs 4 - 7. For example, while Trigger Input 5 exceeds the threshhold set by the user; then the BUFFERED TRIGGER 5 will be active (Ø V.). They are not the equivalent to the Trigger Flags which remain set until cleared by program.

C.8 GPR Input Connector

The GPR Input Connector is JØØ1.
A Cannon 2DA31P is required for User Interface.

Connector Number	Signal Name
1	DATAIN Ø*
2	DATAIN 1*
3	DATAIN 2*
4	DATAIN 3*
5	DATAIN 4*
6	DATAIN 5*
7	DATAIN 6*
8	DATAIN 7*
9	DATAIN 8*
10	DATAIN 9*
11	DATAIN 10*
12	DATAIN 11*
13	DATAIN 12*
14	DATAIN 13*
15	DATAIN 14*
16	DATAIN 15*
17	GPR BUSY
18	GPR DONE
19	USER DEVICE COMPLETE*
20	Not Used
21	Logic Ground
22	+5 Volt Power Flag
23	USER STROBE*
24	USER COUNT*
25	REGRST*
26	GPR DIC
27	Not Used
28	GPR C Pulse
29	GPR S Pulse
30	GPR P Pulse
31	Not Used

C.9 GPR Output Connector

The GPR Out Connector is JØØ2.
A Cannon 2DA31P is required for user interfacing.

Connector Number	In/Out	Signal Name
1	0	DATAOUT Ø*
2	0	DATAOUT 1*
3	0	DATAOUT 2*
4	0	DATAOUT 3*
5	0	DATAOUT 4*
6	0	DATAOUT 5*
7	0	DATAOUT 5*
8	0	DATAOUT 7*
9	0	DATAOUT 8*
10	0	DATAOUT 9*
11	0	DATAOUT 10*
12	0	DATAOUT 11*
13	0	DATAOUT 12*
14	0	DATAOUT 13*
15	0	DATAOUT 14*
16	0	DATAOUT 15*
17	0	GPR BUSY
18	0	GPR DONE
19	I	USER DEVICE COM*
20	-	Not Used
21	-	Logic Ground
22	0	+5 Power Flag
23	-	Not Used
24	0	GPR DOB
25	0	GPR DOC
26	-	Not Used
27	-	Not Used
28	0	GPR C PULSE
29	0	GPR S PULSE
30	0	GPR P PULSE
31	-	Not Used

C.10 TRG Connector

The TRG connector is JØØ6.
A Cannon DAC15P is required for user interface.

Connector Number	In/Out	Signal Name
1	-	Not Used
2	-	Not Used
3	-	Not Used
4	0	TRIGGER 4*
5	0	TRIGGER 5*
6	0	TRIGGER 6*
7	0	TRIGGER 7*
8	0	+5 Volt Power Flag
9	-	Logic Ground
10	I	DISP EXT STROBE*
11	I	AIN EXT STROBE*
12	I	AOUT EXT STROBE*
13	I	BHRC EXT CLK*
14	I	BHRC EXT END*
15	-	Logic Ground

C.11 AIN Connector

The AIN Connector is JØØ8.
A Cannon 2DA31P is required for User Interface.

Connector Number	In/Out	Signal Name
1	I	Analog Channel 8
2	-	Ground for Above
3	I	Analog Channel 9
4	-	Ground for above
5	I	Analog Channel 10
6	-	Ground for above
7	I	Analog Channel 11
8	-	Ground for above
9	-	Logic Ground
10	I	AIN EXT STROBE*
11-21	-	Not Used
22	I	Analog Channel 12
23	-	Ground for above
24	I	Analog Channel 13
25	-	Ground for above
26	I	Analog Channel 14
27	-	Ground for Above
28	I	Analog Channel 15
29	-	Ground for Above
30	-	Logic Ground
31	0	+5 Volt Power Flag

C.12 AOUT Connector

The Aout Connector is JØØ9.
A Cannon DAC15P is required for user interface.

Connector Number	In/Out	Signal Name
1	O	Analog Channel A Output
2	-	Ground for Above
3	-	Ground for Channel X
4	-	Analog Channel X Output
5	-	Logic Ground
6	-	Not Used
7	O	+5 Volt Power Flag
8	I	AOUT EXT STROBE*
9	O	Analog Channel B Output
10	-	Ground for Channel B
11	-	Ground for Channel Y
12	O	Analog Channel Y Output
13	-	Logic Ground
14	-	Not Used
15	I	DISP NEXT STROBE*

C.13 DISP Connectors

The channel Ø DISP connector is JØ1Ø.
The Channel 1 DISP connector is JØ11.
The Channel 2 DISP connector is JØ12.
The Channel 3 DISP connector is JØ13.

A Cannon 2DA31P is required for user interface.

Connector Number	In/Out	Signal Name
1	-	Ground for Channel X
2	0	Channel Y
3	-	Ground for Channel Y
4-5	-	Not Used
6	I	XY EXTEND*
7	0	XY SELECT
8-9	-	Not Used
10	0	XY OPTION ENABLE
11	0	Channel X
12	-	Ground for Channel X
13	-	Ground for Channel Y
14-19	-	Not Used
20	-	Logic Ground
21	-	Logic Ground
22	-	Ground for Channel X
23	0	Ground for Channel X
24	-	Ground for XY INTEN
25	0	XY INTEN*
26	-	Logic Ground
27	-	Logic Ground
28	-	Not Used
29	I	DISP EXT STROBE*
30	I	XY WAIT*
31	0	+5 Volt Power Flag

C.14 Connector Hardware and Tools

CONNECTOR SHELLS AND HARDWARE

Connector	Metal Shell	Plastic Shell	Sliding Lock Retainer	Replacement Contacts
2DA31P	DA24658	DA51210-1	DA51220-1	031-9540-000
DAC15P	DA24658	DA51210-1	DA51220-1	030-1952-000

TOOLS FOR CONNECTORS

Connector	Crimp Tool	Insertion/ Extraction Tool
2DA31P	CCT-CTA	CIET-CTA-2
DAC15P	MS3191-1 with locator P-20-3191-37	CIET-20HDB (expendable, order several)

SUBSTITUTE CONNECTORS

A solder version of the DAC15P connector is available under the part number DA-15P (Cannon or Cinch).

CBL/NOVA RTBASIC

USER MANUAL AND PROGRAMMING GUIDE

April 1973

B. Gaiman
J. B. Walters

Bioengineering Group
Communications Biophysics Group
Research Laboratory of Electronics
Massachusetts Institute of Technology

Copyright 1973
All Rights Reserved

PRECEDING PAGE BLANK NOT FILMED

CBL/NOVA RTBASIC

USER MANUAL AND PROGRAMMING GUIDE

April 1973

B. Gaiman
J. B. Walters

Bioengineering Group
Communications Biophysics Group
Research Laboratory of Electronics
Massachusetts Institute of Technology

Copyright 1973
All Rights Reserved

TABLE OF CONTENTS

Introduction 1

Operation.....2

Description of RTBASIC statements.....3

I/O.....4
General Purpose Digital Register.....11
Real Time Clock.....15
Character Generator.....16
CRT Displays and Plotter.....17
Digital to Analog Output Channels..... 21
Signal Processing Functions.....23
Analog to Digital Convertor.....25
Multi-level Interrupt System.....28
User Subroutines CALL statement.....30

Appendix A.....32

Writing Assembly Language Subroutines for RTBASIC 32
Example of an RTBASIC Subroutine.....45

Appendix B.....48
Short Summary of RTBASIC Statements.....48

Appendix C.....50

RTBASIC Error Messages.....50

Appendix D.....53

RTBASIC Keyboard Commands.....53

Appendix E.....54

Programming Tips for RTBASIC.....54

INTRODUCTION

RTBASIC is an augmented version of the BASIC Programming language which allows the user to control the operation of hardware connected to the Nova to process signals and display them in real time. These additions give the user flexible control over the hardware without complex machine-language programming in the same manner that arithmetic operations can be performed in BASIC. With RTBASIC, a user can sample analog signals, generate analog signals, use a real-time clock, and a general purpose digital register. Signals can be displayed on a CRT or on an X-Y plotter. Interrupts can be generated by externally supplied signals which will cause transfers of control in RTBASIC. Included are signal processing tools, such as a Fast Fourier Transform (FFT).

Features have also been included in RTBASIC to make programming easier. The disk and all devices implemented in the operating system can be used for I/O, and RTBASIC programs and PRINT statements can be displayed on the X-Y displays. The CALL statement has been added in RTBASIC to enable the user to call assembly language subroutines from a RTBASIC program.

OPERATION

RTBASIC exists as a save file on the operating system. In order to run RTBASIC the user types:

RTBASIC (carriage return)

The system will then respond:

RTBASIC (date) (operating system)

Where (date) is the date of the last revision, and (operating system) is the specific operating system on which the version of RTBASIC runs, after this message, the user can proceed to enter and run RTBASIC programs.

The I/O section of this manual describes how to store and retrieve data and programs on the operating system.

DESCRIPTION OF RTBASIC STATEMENTS

The reader should be familiar with the BASIC language and Data General's BASIC interpreter.* The following descriptions contain the format of the statement, the types of arguments allowed, and the results of executing the statement. The following notation is used to describe statement format:

|A| |B|	indicates a choice between A or B in a statement and,
[C]	indicates the optional presence of C in a statement.

For example:

|A| |B|	[,C]

would allow:

A
B
A,C
B,C

in a legal statement. The symbols A, B, and C may be expressions, strings, simple variables and arrays. These are described and defined in Data General's BASIC Manual.

The short summary of RTBASIC statements comprising Appendix B contains examples of actual statements.

---

*Single User BASIC, 093-000042-00, Data General Corp., 1970.

Several statements have been added and modifications have been made to allow RTBASIC to run on the disk operating system. These changes enable the user to store and retrieve RTBASIC programs (which exist as ASCII files on the operating system), to print and input ASCII data files, and to write and read arrays as random records.

The following changes have been made to increase compatibility with the operating system:

to delete the previous character typed, the rubout key is used and a ← (back arrow) is echoed on the terminal;

in order to delete the current line of input, a shift L is typed and a \ (back slash) is echoed;

the keys CNTRL A and CNTRL C will cause the appropriate breaks according to the specific operating system.

The following are new RTBASIC statements:

OPN

The OPN statement allows a file or device to be opened for use with the LIST, PRINT INPUT or RAND statements. OPN can be used either as a keyboard command or in a program statement.

OPN "| File name / device |" , n

will open the file (named filename) or device (e.g. $PTR or $LPT) on channel n (n is an expression with a value of 0 thru 7). If the file does not exist, it will be created.

CLS

The CLS statement closes a file or device.

CLS n

closes the file associated with channel n. This statement can be used a keyboard command or as a program statement.

PRG

The PRG command is used to read a previously saved RTBASIC program from a file or device in the following format:

PRG " | Filename / Device | "

If another program was running previously, a NEW command should be issued before PRG. The NEW command will delete all program statements. The PRG statement can also be used in a program. When it is executed in a program statement, PRG will overlay RTBASIC program statements or call in additional RTBASIC statements while still running since PRG does not automatically do a NEW command. However, if PRG is used to overlay program statements, care should be taken so the line numbers of the new statements and the statements being overlaid match.

RAND

This statement allows an array to be read from or written into a random file in binary form. The format of the statement is as follows:

RAND #CHN n,A,R, | "R" / "W" |

In this statement, n is the channel number of a previously opened file, A is the array, R represents the initial record number, and the last argument, "W" or "R", specifies whether the array is to be written ("W") into the file or read ("R") from the file.

The minimum size array which can be used in the RAND statement is 32 words (DIM A(31)), and the array size should be a multiple of 32, which is the size of a record. This means that in the case of an array of size 128, the initial record number would have to be incremented by 4 each time the array (presumably containing new information) was rewritten on the random file.

DOS

The proper return to the operating system is made by giving the keyboard command -

```
DOS
```

which will close all files and return.

I/O Extensions to Single-User BASIC

The LIST, PRINT, and INPUT statements have been extended to facilitate the use of ASCII files on the operating system.

LIST

LIST #n or XXXLIST #n causes the output of the normal BASIC LIST command to be directed to a file or device. N is the channel number associated with the file or device by an OPN statement.

PRINT

PRINT CHN n, is equivalent to a BASIC PRINT statement, however, the output is directed to a file or device which has previously been opened on Channel n. A # can be used as a shorthand for CHN. The output is stored as ASCII bytes.

INPUT

INPUT CHN n, is equivalent to a BASIC INPUT statement with the input taken from the file or device associated with channel N. A# can be used as a shorthand for CHN. The input file should consist of ASCII data with each number terminated by a carriage return. In order to reuse the data in a file as input, the file must be closed (CLS n) and then reopened (OPN "Filename", n).

Example 1 shows how RTBASIC is started and how a program can be created, run, stored and finally recalled.

The first step is to bootstrap the Operating System. RTBASIC is started in Step 2 by typing RTBASIC. Steps 3 and 4 show the program typed in and run. After the program has been run and the user wishes to save it, he first opens a new file on an unused channel (Step 5), lists the program on that channel (Step 6), and finally closes the channel (Step 7), thereby closing the file. The user can return to the operating system by typing DOS; the system will respond with an R as in Step 7.

RTBASIC is then started again (Step 8), and the program previously saved is recalled with the PRG command (Step 9). In Steps 10 and 11, the program is listed and run again.

BOOTSTRAPPING THE DISK

STEP ONE

1. Turn NOVA Power Switch to "ON" (Not Lock) at least one lamp on the front panel should come on when you do this.

2. Trip the "RESET" Switch a few times.

3. Turn on the teletype by rotating the knob at the right under the keyboard counterclockwise.

4. Turn on the disk, by lifting up the switch in the red panel under the NOVA. As the disk comes up to speed, it makes a little noise--do not worry unless it is more than you heard the last time. A distinctive klunck will be heard when the disk heads are lowered. After this, you may continue the startup procedure.

5. Set the front panel switches to 376 (all numbers in octal).

6. Trip the examine switch.

7. Set front panel switches to 060120.

8. Trip the "DEPOSIT" switch.

9 Set the front panel switches to 377.

10. Trip the "DEPOSIT NEXT" switch.

11. Set the front panel switches to 376.

12. Trip the "START" switch.

13. Within a few seconds the teletype should type out a message identifying the operating system on the disk. Then the NOVA will halt. (The NOVA is halted if the RUN lamp is off.)

14. Trip the CONTINUE switch. The teletype will respond with an "R" within a few seconds. This indicates that the command line interpreter is ready to receive user commands.

EXAMPLE 1

```
R
RTBASIC
RTBASIC 3/17/73 DOS04

10 FOR I=0 TO 5
20 ;SQR (I)
30 NEXT I
40 END

RUN
 0
 1
 1.41421
 1.73205
 2
 2.23607

OPN "BPROG",4
LIST #4
CLS 4

DOS

R
RTBASIC
RTBASIC 3/17/73 DOS04

FRG "BPROG"

LIST
10   FOR I= 0 TO 5
20     PRINT  SQR (I)
30   NEXT I
40   END

RUN
 0
 1
 1.41421
 1.73205
 2
 2.23607

DOS
```

2) Start RTBASIC

3) Input Program

4) Run Program

5) Open Disk File to store program
6) List program on Disk File
   Close

7) Return to Disk Operating System

8) Start RTBasic

9) Read in program from Disk File BPROG

10) List Program

11) Run Program

GENERAL PURPOSE DIGITAL REGISTER

OGPR

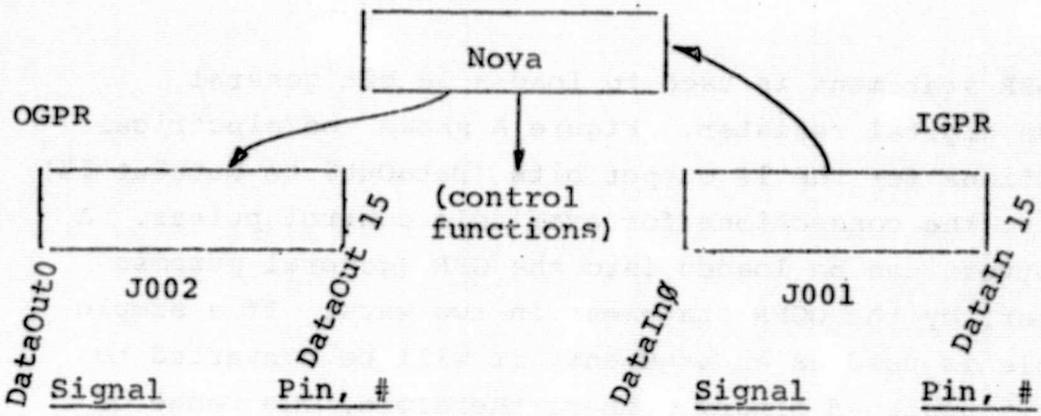
The OGPR statement is used to load a 16 bit general purpose digital register. Figure A shows the electrical connections for the 16 output bits (DataOut0 to DataOut 15) and also the connections for available control pulses. A bit pattern can be loaded into the GPR (general purpose register) by the OGPR statement in two ways. If a simple variable is used as an argument, it will be converted to a 16 bit unsigned binary number, therefore, the range of acceptable values is 0 to 32,768. If the value of the argument is outside this range, an error message will be returned. For example:

```
10 LET A1 = 65
20 OGPR A1
```

will cause the following binary bit pattern to be loaded into the GPR:

DataOutØ	DataOut1	DataOut2													DataOut15	Data output line
0	0	0	0	0	0	0	0	1	0	0	0	0	0	1		logical value

It is also possible to use an array as an argument in an OGPR statement. In this case, each element in the array (which must have a dimension of at least 16) will load a corresponding bit in the GPR with a 1 if the value of the element is non-zero; otherwise, a zero will be loaded into the GPR. The first element in the array will load the first bit, DataOutØ, and elements in the array after 16 will be ignored. If the following array, A, is used to



Signal	Pin, #	Signal	Pin, #
DataOut Ø	1	DataIn Ø	1
DataOut 1	2	DataIn 1	2
DataOut 2	3	DataIn 2	3
DataOut 3	4	DataIn 3	4
DataOut 4	5	DataIn 4	5
DataOut 5	6	DataIn 5	6
DataOut 6	7	DataIn 6	7
DataOut 7	8	DataIn 7	8
DataOut 8	9	DataIn 8	9
DataOut 9	10	DataIn 9	10
DataOut 10	11	DataIn 10	11
DataOut 11	12	DataIn 11	12
DataOut 12	13	DataIn 12	13
DataOut 13	14	DataIn 13	14
DataOut 14	15	DataIn 14	15
DataOut 15	16	DataIn 15	16
GND	21	GND	21
+5	22	+5	22
C	28	C	28
S	29	S	29
P	30	P	30
DATOB·DEVSEL	24		
DATOC·DEVSEL	25		

Figure A. Diagram and electrical connections of general purpose digital register.

load the GPR:

```
A(0) = .5
A(1) = 0
A(2) = 1
A(3) = 1510
A(4) = 0
A(5) = 0
A(6) = 0
A(7) = 0
A(8) = 0
A(9) = 0
A(10) = 0
A(11) = 1
A(12) = 2
A(13) = 3
A(14) = 4
A(15) = 5
```

The following binary bit pattern will be loaded into the GPR:

```
DataOutØ  DataOut1  DataOut2                          DataOut15
1 0  1 1 0 0 0 0 0 0 0 1 1 1 1 1
```

The format of the OGPR statement is:

```
OGPR |simple |variable
     |       |       [,"control function"]
     |array  |
```

The control function, which can be an S, C, or P lets the user generate one of the three available pulses. The electrical connections for these pulses are shown in Figure A. The register will be loaded within about one millisecond after the execution of a GPRO statement is started, and the control pulse (if S, C, or P is present) will occur about 200 microseconds after the register is loaded.*

* Times are for the old NOVA Computer.

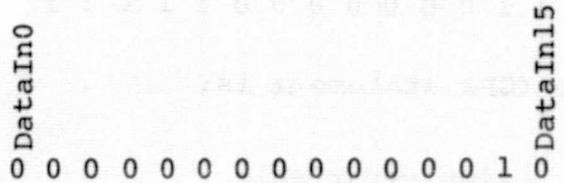
IGPR

This statement is similar to the OGPR statement; it is used to examine a 16 bit binary input (DataInØ to DataIn15) and to load this pattern into either a simple variable or into the elements of an array. The input must be placed on the DataIn lines while the IGPR statement is being executed (execution time is approximately one millisecond). The pin connections for DataInØ to DataIn15 are shown in Figure A.

The format of this statement is:

IGPR [simple-variable / array] [,"control function]

The presence of the control function has the same effect that it has in the OGPR statement which is the generation of a pulse. If the binary bit pattern was the following:



and the argument of the IGPR statement was a simple variable, after the statement was executed, the simple variable would have the value 2. If the argument was an array named B, elements B(Ø) to B(13) and B(15) would be set to zero, and B(14) would be set to 1. All elements after B(15) would remain unchanged.

Real Time Clock

CLK

The CLK statement is used to start, to interrogate, and to stop the real time clock. The result of the interrogation is the number of clock cycles that have occurred since the clock was last started. The period can be set and the clock started by executing a CLK "Sn" statement where n is a number from 0 to 3. The period is set by n according to the following table:

n	Period
0	60 Hz
1	10 Hz
2	100 Hz
3	1000 Hz

The format and operation of the CLK statement is:

CLK	simple variable	Load current value of clock into simple variable.
	"S"	Reset and start clock (60 Hz.)
	"Sn"	Reset and start clock and set period.
	"D"	Stop clock (DONE)

If no period is specified, a clock frequency of 60 Hz. will be used. The clock is accurate to within one millisecond of the true time elapsed after a clock start is executed.* The clock can be run for approximately 25 days at the highest frequency (1000 Hz) without overflowing.

*The accuracy of the clock will be adversely affected by YPLOT or BPLOT of arrays on the X-Y Plotter if the clock rate is set at 1000 Hz.

CHARACTER GENERATOR

WRITE

The WRITE statement is identical to the PRINT statement in BASIC except that the output device is a CRT display or the X-Y plotter. Since it is desired that the initial location of the displayed text be any region of the display, the horizontal and vertical position of the initial point of the output of the WRITE statement can be specified by using a PLOT statement. The output devices, either a CRT, an X-Y plotter, or both, may be selected by using the XYSEL statement.

DSPLY

This statement is similar to the LIST statement in BASIC. The output device is a CRT display and 20 lines of the program are displayed at a time. The 20 lines which are displayed are specified by typing nnnnDISPLY where nnnn is the initial line in the nnnnLIST statement. The 20 lines will be displayed until the ESC key is struck, and then control will be returned to the user.

CRT DISPLAYS AND PLOTTER

XYSEL

XYSEL lets the user select which devices will be used for displays in conjunction with the PLOT, YPLOT, BPLOT, and WRITE statements. The format of the XYSEL statement is:

XYSEL " {device list} |S / X| "

The device list lets the user specify which devices are to be used. The table below lists the devices available and the letters used to represent them.

Device	Representation
Tektronix 602 Scope (mounted in cart)	A
HP Display	B
HP X-Y Plotter	P

Inclusion of an S with a device representation in the XYSEL statement causes points to be intensified (or plotted) without a device list, the display is positioned without any output. When executing a WRITE statement, this feature is useful for moving the X-Y coordinates without actually plotting or intensifying the point. The inclusion of an X specifies that an external signal will initiate both the positioning and intensification (or plotting). If no device is given in the statement, then there is no intensification (or plotting) upon executing PLOT or BPLOT statements.

The following are two examples of XYSEL statements:

XYSEL "ABPS"	Use all devices and output the point
XYSEL "S"	Do not plot any points; just move the CRT beam or pen.

PLOT

The PLOT statement positions, the beam of the CRT or the pen on the plotter according to a pair of arguments. The specified point will be intensified (or plotted) if an S and device list was placed in the preceding executed XYSEL statement. The arguments in

```
PLOT X,Y
```

can be variables or any expression. Thus,

```
PLOT I,SIN(Z*2)
```

is a legal statement. Arguments ranging from -512 to 511, -512 to 511 will cover the entire scale of the output devices. The point 0,0 is in the center of the display with 511,511 corresponding to the upper righthand corner.

YPLOT

The YPLOT statement is used for plotting a sequence of points stored in the form of an array. The array specifies the abscisa. The ordinate is incremented by an adjustable step value automatically by RTBASIC. If no step is specified, the ordinate is incremented by one. The data sequence is started at the extreme left hand position (corresponding to x = -512) on the selected output devices. The format of the YPLOT statement is:

```
YPLOT array [, step]
```

the step may be a variable or any expression; and if it causes the ordinate value to exceed 512, wrap-around will occur in the display.

The statement

```
YPLOT A
```

where A is a 10 element array produces the same display as execution of the following three statement program using only the PLOT statement:

```
10 FOR I  = 0 TO 9
20 PLOT I-512,A(I)
30 NEXT I
```

and

```
YPLOT A,5
```

is similar to:

```
10 FOR I  = 0 TO 9
20 PLOT I*5-512,A(I)
30 NEXT I
```

However, in both cases, YPLOT offers a better display because of its increased speed in plotting and lower storage requirement. With the YPLOT statement up to 1024 points can be displayed on a CRT without flicker.

BPLOT

The BPLOT statement will plot points with X and Y coordinates taken from two arrays (which must have identical dimensions). The format of a BPLOT statement is:

```
BPLOT array, array
```

where the first array contains the X values and the second contains the Y values.

PEN

The PEN statement is used to raise and lower the pen on the X-Y plotter. If the argument after PEN is "D", the pen will be lowered, and the pen will be raised if the argument is "U". The plotter need not be selected for the pen to be raised and lowered. The PEN statement has no effect when the null point plotter is installed. The null point plotter must be installed if the user wishes to write characters on the plotter with a WRITE statement.

DIGITAL TO ANALOG OUTPUT CHANNELS

ADA,BDA,DAB

These statements control the two D-A convertors (A,B) which are used for analog outputs. The ADA and BDA statements allow the user to generate a single analog value if a simple variable is specified as an argument or a sequence of values (the timing controlled by an internal clock or an external signal) if an array is specified as an argument. The format of these statements are:

```
DA      | simple variable                                  |
or      | array           | ,conversion frequency          |
DB      |                 | ,"X"                           |
```

If a simple variable with a value of -512 to 511 is specified as an argument, an analog voltage between -2 to 2 volts will appear on either the A or B channel.* If an array is specified as the argument, then a timing source must be specified which will initiate the conversion and step through the elements of the array. An "X" selects an external source*, otherwise an expression specifies the frequency of conversion. The time between conversions is specified in tenths of milliseconds with the highest frequency being 2000 values/sec (equals 1/.5 milliseonds). The output will remain at the value of the last element in the array until a new statement is executed which uses the D-A convertors.

---

*See the CBG Nova System Reference Manual for the electrical connections of these two channels.

The format of the DAB statement is:

```
DAB  | simple variable, simple variable                    |
     | array, array     |conversion period|               |
     |                  |",X"             |               |
       A channel          B channel
```

This statement is similar to the DA and DB statements, however, it permits output on both the A and B channels at the same time. The two arrays specified are required to have the same number of elements.

Since the D to A statements can run under the interrupt system, the user should be familiar with the WAIT statement.

SIGNAL PROCESSING FUNCTIONS

FFT

The FFT statement is used to execute a Fast Fourier Transform. A complex transform is used when two (equal-dimensions) arrays are specified. If only one array is specified, a real transform will be taken and only the "positive" frequencies will be returned as results. An"R" after the arrays results in a reverse transform. The following examples show the effect of the different FFT statements:

```
10 DIM A (255)
20 DIM B (255)
    .
    .
    .
100 FFT A,B          take the complex transform
                     of the array A +jB.  The
                     results are placed in A,B.

120 FFT A,"R"        Take the reverse real
                     transform of A.  Return with

                     cos coefficients in A(0)-A(N/2-1)
                     sin coefficients in A(N/2)-A(N-1)
```

The format of an FFT statement is:

```
FFT | Array, Array |
    |              | [,"R"]
    | Array        |
```

The dimensions of all arrays must be a power of two. Since the FFT computes the discrete Fourier Transform which is

$$F\left(\frac{K2\pi}{NT}\right) = \sum_{n=0}^{N-1} F(nT)\, e^{\frac{-j2\pi nK}{N}}$$

itself a periodic function, the "negative" frequencies in a transform containing N points will appear as the output of the FFT from N/2 to N. If the sampling rate is $2\pi/T$ and the number of samples is N, then (N/2-1) the point of the FFT will correspond to a frequency of π/T and adjacent points (the frequency resolution) will be separated by $2\pi/NT$.

SAMP

The SAMP statement is used to sample analog signals by allowing the user to: select the number of samples he wishes to take; select which of sixteen multiplexed A-D channels will be used; and to determine the frequency at which samples are taken. Also, the user can specify that samples are to be continually taken and placed in a buffer array until a preset trigger level is exceded. When this level is exceded, the current samples in the buffer array are placed in a storage array, and sampling is continued until the rest of the storage array is filled with the new samples. Both the size of the buffer array and the storage array are adjustable. This allows the data immediately preceding a detected event to be examined.

The format of the first SAMP statement is:

```
SAMP  |simple variable|  ,channel#[,frequency]
      |array            |
```

The channel #, frequency, and trigger level can be any legal BASIC expression. The total number of samples to be taken is determined by the size of the storage array. A simple variable is used if only one sample is wanted. If only one sample is to be taken, the frequency need not be specified.

There are sixteen A-D channels available; the A-D convertor is preceded by a sample and hold circuit with a 10 nanosecond aperture. The input range is -2 to +2 volts.

Channels 4 through 7 have adjustable gain, adjustable offset voltages, and may be AC-coupled by means of front-panel controls. The maximum positive analog signal is converted to $+511_{10}$ while the maximum negative signal is represented as -512_{10} in RTBASIC. Channels 0 through 3 are permanently connected to potentiometers on the front panel. The sampling frequency is specified by giving the interval between samples in tenths of milliseconds with the smallest period being .2 milliseconds. The following example shows the use of the first type of SAMP statement:

```
10  DIM A (99)
20 SAMP A,7,10
```

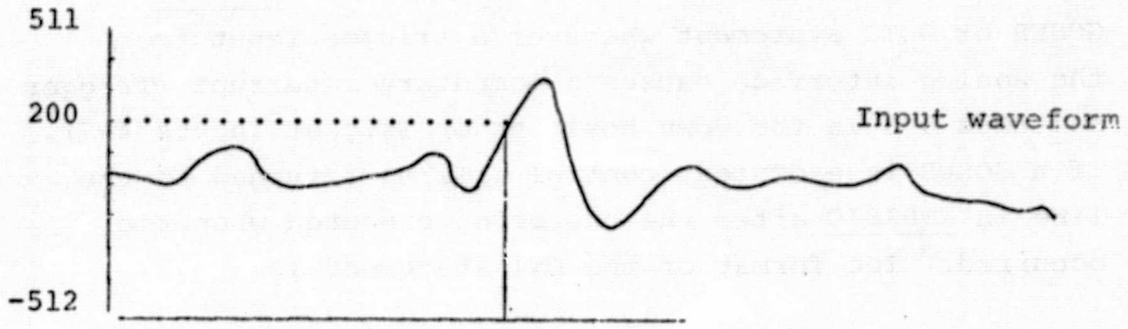
this program takes 100 samples from channel 7 sampled at 1KHz and stores them in Array A.

The second sampling statement has the format

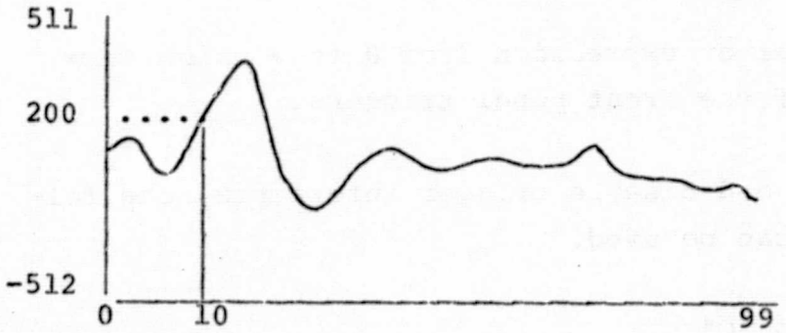
```
SAMP     Storage Array    ,channel #,frequency
                          ,trigger level [buff. array]
```

and is capable of collecting data before and after a specified event occurs. In this SAMP statement, the optional buffer array serves as temporary storage for the sampled data and it also determiens the number of samples before the trigger level is exceded which will be saved and loaded into the final storage array. For wxample, if the dimension of the buffer array is N then M samples will be placed into the storage array, N of these samples will have been taken before the trigger level was exceded and M-N samples will have occurred after the triggering event. The trigger level is specified from -512 to 511 corresponding to -2 to +2 volts. The following example shows the use of the trigger:

```
10 DIM A(99)
20 DIM B(9)
30 SAMP A, 7, 2, 200, B
```



Input into buffer B



Array A when sampling is finished

Array B is used for temporary storage. When the input on channel 7 excedes (200/511)*2 volts, the samples in B (10 samples before the trigger was exceded) will be placed into array A, and the other 90 elements of A will be filled with the samples taken immediately after the trigger level was exceded.

Since the SAMP statement runs under the interrupt system, the user should be familiar with the WAIT statement.

MULTI-LEVEL INTERRUPT SYSTEM

ONI

The ONI statment allows the execution of an RTBASIC GOSUB or GOTO statement whenever a trigger input from the analog interface causes a momentary interrupt (Trigger switches 0-3 in the down position or Trigger inputs 4-7). If a GOSUB is executed, control will be returned to the line in RTBASIC after the one being executed when the occurred. The format of the ONI statement is:

```
1Ø   ONI   n
11  {GOTO line number}
    {GOSUB line number}
```

where n is a number or expression from Ø to 7 which corresponds to one of the front panel triggers.

In order to clear and disable trigger interrupts, the following statement can be used.

```
ONI "C"
```

This will disable the effect of any ONI statement previously executed until another ONI is issued. This statement is useful to clear interrupts when continuous trigger inputs are coming into channels 4 through 7.

Caution - If the condition of ONI contains a GOTO or GOSUB which branches to a line # which doesn't exist, then it is possible to get an ERR 13 at a line number which doesn't contain a GOTO or GOSUB. The ERR 13 is generated when the improper ONI is executed and the error is assigned to the line which was interrupted by the ONI trigger.

WAIT

The WAIT statement is used in conjunction with the SAMP statement and D/A statements (ADA, BDA, and DAB) if the user decides that he must wait for sampling or analog output to be completed. If a WAIT statement is used, the execution of a RTBASIC statement is suspended until the SAMP or D/A statement signals that it is finished.

WAIT will cause the machine to wait until the SAMP or D/A statement is finished before letting RTBASIC begin execution of the next line.

The SAMP statement and D to A statements also have an automatic WAIT built in (a separate one for SAMP and one for ADA, ADB, and DAB). The following two sequences have almost identical action:

```
10 SAMP A, 4, 10
20 WAIT
30 SAMP B, 4, 10
```

```
10 SAMP A, 4, 10
20 SAMP B, 4, 10
```

In the first set, the sample statement at 10 will be executed and the program will wait at 20 until all of array A is filled with samples then the statement at 30 will be executed and samples placed in B. The second program does the same things execpt there is no delay between the end of sampling array A and the beginning of sampling array B.

Since the SAMP statement and the D/A statements all use the High Resolution Clock (HRC), care should be used to either use the WAIT statement or the same clock frequencies if the SAMP statement has any possibility of overlapping with a D/A statement. A WAIT statement is not necessary when a single value is sampled (SAMP A1,Ø).

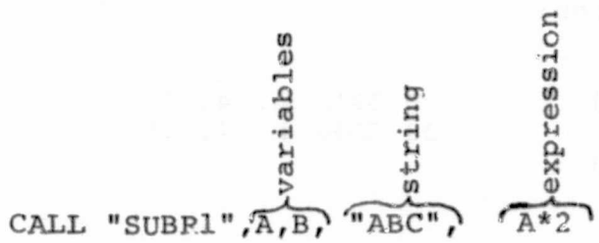
USER SUBROUTINES

CALL

The CALL statement allows the user to write assembly language subroutines and call them from RTBASIC. The format of a CALL statement is:

CALL "SUBRn" [,list of arguments]

where n is a number from 1 to 5, and the list of arguments contains any number of strings, variables, and expressions separated by commas. Thus, the user can call a subroutine which he has designated with a number from 1 to 5. Arguments may be passed in CALL statements in the form of variables, strings, or expressions in any order. For example:



Appendix A contains procedural and programming information important in writing subroutines. When the CALL statement is used, RTBASIC checks to see if the routine being called is actually loaded; and if it has been loaded in the proper manner and location (see loading procedures for subroutines below). If RTBASIC determines that the subroutine does not exist or has been improperly loaded, it will return the appropriate error message. Caution should be used in writing subroutines because improper handling of control or returns could cause a fatal error in RTBASIC, and this would necessitate reloading after correcting the cause of the error.

LOADING PROCEDURES FOR SUBROUTINES

Subroutines must be written in relocatable form, hence, the user should know how to use the relocating assembler to write them*. To load subroutines into RTBASIC, the follwoing DOS RLDR command is used:

```
RLDR USERBASIC/S @RTB1@  USERSUB1.RB [USERSUB2.RB...5.R3]
        RTB2.RB USERBASIC/S
```

USERSUBn.RB is the relocatable binary of the user's subroutine(up to /5 user subroutines may be loaded). USERBASIC will be a savefile of RTBASIC and the added subroutines which can be executed by typing USERBASIC. The combined length of all user subroutines should not exceed 2000_{10} in order to allow room for an average sized RTBASIC program in a 16K memory which might consist of 200 statement lines and storage for a 1000 point array. If storage is exceeded, RTBASIC will return the appropriate error message.

---

*Extended Assembler 093-000040-00, and Relocatable Loader 093-000039-00, Data General Corp., 1969.

C-3

APPENDIX A

WRITING ASSEMBLY LANGUAGE SUBROUTINES FOR RTBASIC

Using the CALL statement, a user can call and execute subroutines written in relocatable assembly language. For a subroutine to be called, it must contain an entry point of the form:

```
.ENT SUBRn
```

in the assembly code. SUBRn should be the label of the starting location of the subroutine. To return from a user subroutine, the following code must be included.

```
.EXTD    .LINE
.
SUBRn:          ;STARTING POINT OF SUBROUTINE
.
.
JMP @ .LINE     ;RETURN TO RTBASIC
```

PASSING AND EVALUATING ARGUMENTS

Also, the user can pass arguments in the CALL statement. These arguments can be strings, variables, or expressions. When arguments are used in a CALL statement, they are converted into a numerical representation. The following section discusses this format and the format of floating-point numbers in RTBASIC. A list is then given of the subroutines in RTBASIC available to the user which can be used followed by an actual user subroutine program.

The arguments in a CALL statement, such as:

```
CALL "SUBR2", "ABC", X*2,Z
```

are converted to octal numbers and stored in a byte (8 bits) format (listed in Table 1). Numbers are represented as a double word RTBASIC floating-point constant. The above list of arguments would be converted to:

Constants			Second Byte	First Byte
0	7,8	15		
301	351	← PBC	A	"
303	302		C	B
361	000		,	end of string
373	201		*	symbol table ref. for X
100	000	}	floating point repre-	
000	404	}	sentation for 2	
205	361		symbol table ref for Z	
	342			' end of line

Table: Contents specified by arguments in CALL statement location specified by PBC.

Table 1

Program Byte Formats

Symbol	Representation (Base 8)
(	362
)	366
'	361
"	351
[	363
]	367
+	371
-	370
*	373
/	372
end of line	342
character code (letters and numbers)	ASCII code + 200
Start of string (")	351
end of string (null)	0
symbols	symbol table reference +200
numbers	floating-point format

PBC, the program byte counter, is a pointer in RTBASIC which marks the current program byte. After a subroutine has been called, PBC will point to the first program byte of the argument list. In the example above, PBC would point to the symbol " represented as 351. Numbers in the argument list are represented as double floating-point words; the format is described in Table 2.

In order to use the argument lists passed in a CALL statement, it is necessary to have routine which can access the program bytes, evaluate expressions, and determine the size and locations of arrays and simple variables.

If arithmetic is to be done, the appropriate floating-point routines must be available. Table 3 contains a list and short description of these routines.

If the user had a subroutine which required an argument having a string, then an expression, and finally a simple variable or an array, he would proceed in the following manner to manipulate the argument list. The PBC will be pointing to the first argument in the list so that the user need not know explicitly where the first argument is. To obtain the first program byte, he calls ACNB by executing:

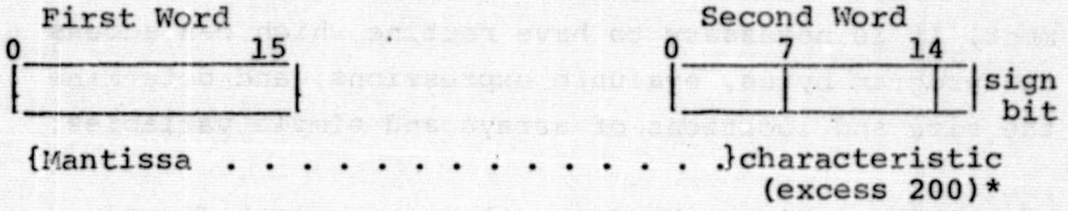
```
JSR ACNB
```

If the first argument is actually a string, AC2 should contain a 351 (the representation for the symbol "). If it is in fact a string, he can get the first character by again calling ACNB.

```
JSR ACNB
```

Table 2

Floating-Point Format



First Word Second Word (Bits 0 to 7)	24 Bit Mantissa
Second Word (Bits 8-14)	characteristic $+200_8$
Second Word (Bit 15)	Sign Bit 1 negative 0 positive

*The characteristic is in excess 200 notation (i.e., 200_8 added to the regular floating-point characteristic).

TABLE 3

List of RTBASIC Argument Handling Routines

Name	Function
.ADDF	Floating-point add
.SUBF	Floating-point subtract
.MULF	Floating-point multiply
.DIVF	Floating-point divide
.FLO/T	Convert integer to flt. pt.
.FIX	Convert flt. pt. to integer
ACNB	Access next program byte
.EVEX	Evaluate expression
.LINE	Return to RTBASIC
.ERRA	Print error, return to RTBASIC
ALOC	Allocate storage
.LUVD	Look up variable definition
.ADIM	Get array subscripts
ARRS	Get array size

Every routine with a . is called by a JSR @__. The appropriate .EXTN pseudo-ops must be included in the assembly code.

The remaining characters in the string can be obtained in the same way. A 0 will appear in AC2 after ACNB has been called when the end of a string has been reached. If the user is now looking for an expression, he will again call ACNB to find the comma (361) in AC2 which separates the string and the expression. If he wants to evaluate the expression, he calls the expression evaluator by executing a

```
JSR  @.EVEX
```

the result will appear in AC0 and AC1 in floating point format. If the expression evaluator detects an error, AC2 will contain a 0 so it is the usual procedure to check AC2 after calling the expression evaluator. The expression evaluator will place the PBC pointing after the comma if a expression is terminated by a comma as part of a list. (If at any time the user wants to check for the end of the line, he should call ACNB and check for a 342 in AC2.) After evaluating the expression, the user wishes to check for the variable which is supposed to be in the argument list so he calls ARRSZ. The user can find out information about the variable by executing the following code:

```
.EXTN   ARRSZ
    .
    .
JSR @ADRSZ
                        :RETURN HERE IF SIMPLE VARIABLE
                        :RETURN HERE IF ARRAY
    .
    .
    .
ADRSZ:  ARRSZ
```

If the variable was a simple variable, the routine will return to the first location after the JSR, and AC2 will contain the storage address of this simple variable. If the variable was an array then the return will be made to the second location after the JSR. AC2 will contain the address of the first element in the array, and AC1 will contain the total number of elements in the array. The user can, if necessary, back up to or skip over program bytes by incrementing (ISZ) or decrementing (DSZ) the program byte counter (PBC).

Floating-Point Arithmetic Routines

F0 and F1 are floating-point numbers. These routines are called with F0 and AC0 and AC1, and AC2 contains a pointer to F1.

Routine	Computes
.ADDF	F0+F1
.SUBF	F1-F0
.MULF	F0*F1
.DIVF	F1/F0

The result is placed in AC0 and AC1.

Conversion Routines

.FLOT is called with a two's complement signed integer in AC0 (msb) and AC1 (lsb). The result, a floating-point word, is placed in AC0 and AC1.

.FIX is called with a floating-point word in AC0 and AC1 and returns a two's complement signed integer in AC0 and AC1.

Argument Handling Routines

To obtain the current program byte, the following routine is called:

ACNB

This routine is called with PBC pointing to the current program byte. It returns with the current program byte in AC2, and it increments PBC.

The following routines are useful in manipulating variables and evaluating expressions:

.LUVD

.LUVD is called with AC2 containing the symbol reference for a variable (which is the program byte representation of a variable). The routine returns with AC2 containing the core address of the variable if the variable has been defined. If .LUVD returns with a zero in AC2, storage must be allocated with a call to ALOC before the value in AC2 is changed. Bit Ø of ACØ is the array flag for defined variables. If the variable is an array, bit Ø will equal 1 if it is a simple variable bit Ø equals Ø.

ALOC (used after calling .LUVD)

ALOC is called with the following code:

```
.EXTN ALOC
.
.
JSR @ AALOC
.
.
.
AALOC:  ALOC
```

It will allocate storage for the undefined variable in the preceding .LUVD routine, and AC2 will contain the core address of the variable.

.ADIM

This routine is called with AC2 containing the symbol reference for a variable. On return, ACØ contains the integer value of the first subscript and AC1 contains the integer value of the second subscript.

ARRSZ

ARRSZ is called with PBC pointing to a variable. It has two return points as described above. If the variable is an array, AC1 will contain the size of the array (equals [dim 1 + 1]*[dim 2 + 1]), and AC2 will contain the address of the first element in the array. If the PBC is pointing to a simple variable, then ARRSZ will return to the location immediately after the JSR, and AC2 will contain the address of the variable. PBC is incremented once for each call to

ARRSZ.

.EVEX

This routine is the expression evaluator. The format of an expression is described in BASIC, and it is terminated by comma, or a semicolon, or by the end of a line (spaces are ignored).

The evaluator is called with PBC pointing to the first byte of an expression.

The routine returns:

If AC2 equals zero then there was an invalid expression or possibly a string

If AC2 is nonzero then the floating-point value of the expression is in AC0 AC1

AC3 contains the next character (byte) in the program

If the expression evaluator is called pointing to a variable, or a dimensional variable (i.e., A(1,2) then AC2 will contain the storage address of the variable on return.

The following routines are used in returning from user subroutines.

There are two ways in which the user can return to RTBASIC:

JMP @ .LINE

This is the normal return. The PBC will be moved

to the next line and control will be returned to RTBASIC.

$$\text{JSR } \overset{@}{N} \text{ .ERRA} \quad \text{or} \quad \text{JSR } \overset{@}{N} \text{ .ERRE}$$

This is the error return. The value of N will be converted to a decimal value and will be printed on the console as the number or error. The call to .ERRA will also print the line number at which the error occurred, and execution will continue. The call to .ERRE will stop execution.

EXAMPLE OF USER SUBROUTINE

The following program SAMP is an example of a subroutine which can be called from RTBASIC. This program is a simple sampling routine that uses the real time clock to control the sample rate and does not run under interrupts. To execute the subroutine, the following statement is given:

```
CALL SUBR2 A, 7, 2.
```

This statement will call the SAMP program with arguments A, 7, 2. The subroutine first determines the size of array A or determines if A is a simple variable. The number of samples to be taken which is equal to the dimension returned by ARRSZ is stored, and the address of the first element is loaded into ADRBF. The comma in the list of arguments is skipped by incrementing the PBC (program byte counter). The channel number and clock period are determined by evaluating the corresponding expression (the expression evaluator evaluates an expression and leaves the PBC pointing to the byte after the comma). The channel number and clock period are converted to integers, and the I/O instructions are executed to set up the clock and A-D converter correctly.

Sampling is started; and after a sample is taken, the integer sample is converted to a floating point double word and stored in the selected array or simple variable. When the correct number of samples is taken, the subroutine returns to RTBASIC and execution begins again with the next line. If an error was detected by the expression evaluator, this subroutine will return to RTBASIC through the error package which will cause "ERR 67" to be written and execution will be stopped.

```
  0001   SUBR2
01                          .TITL   SUBR2
02
03
04                  ;SUBR2 USES THE RTC TO SAMPLE WAVEFORMS
05                  ;THE USER SPECIFIES THE ARRAY OR SIMPLE VARIABLE
06                  ;TO BE LOADED WITH THE RESULT
07                  ;ALSO THE USER SPECIFIES THE CHANNEL NO.
08                  ;AND THE CLOCK FREQUENCY
09
10
11                  ;USE    CALL "SUBR2",A,C,F      TO EXECUTE
12
13                  .ENT SUBR2
14                  .EXTD   ACNB,C342
15                  .EXTD   .EVEX,.FIX,PBC,.FLOT,.LINE,.ERRA,C1,S1,.PK
16                  .EXTN   ARRS2,SUHRC
17                  .NREL
18
19 00000' 060277 SUBR2:   INTDS   ;DISABLE INTERRUPTS
20 00001' 006457          JSR @ADRSZ
21 00002' 000450          JMP SSAMP          ;SINGLE SAMPLE
22 00003' 044453          STA 1,NSAMP        ;STORE THE NUMBER OF SAMPLES
23 00004' 050453 STADR:   STA 2,ADRBF        ;STORE ADDRESS OF FIRST WORD IN ARRAY
24 00005' 010005$         ISZ PBC            ;GET RID OF COMMA
25 00006' 006003$         JSR @.EVEX         ;GET CHANNEL NO.
26 00007' 151015          MOV# 2,2,SNR       ;CHECK FOR BAD EXPRESSION
27 00010' 000440          JMP ER
28 00011' 006004$         JSR @.FIX          ;CONVERT CHANNEL N0.  TO INTEGER
29 00012' 065031          DOA 1,AIN          ;CHANNEL SELECT REGISTER
30 00013' 102400          SUB 0,0
31 00014' 063031          DOC 0,AIN          ;CONTROL REGISTER (S PULSE MODE)
32 00015' 006003$         JSR @.EVEX         ;GET CLOCK PERIOD
33 00016' 151015          MOV# 2,2,SNR       ;CHECK FOR BAD EXPRESSION
34 00017' 000431          JMP ER
35 00020' 006004$         JSR @.FIX          ;CONVERT PERIOD TO INTEGER
36 00021' 065014          DOA 1,RTC          ;LOAD PERIOD
37
38 00022' 060114 SAM:     NIOS RTC           ;START REAL TIME CLOCK
39 00023' 060131          NIOS AIN           ;START CONVERSION
40 00024' 063531          SKPBZ AIN          ;WAIT FOR CONVERSION
41 00025' 000777          JMP .-1
42 00026' 064431          DIA 1,AIN          ;TAKE IN SAMPLE
43 00027' 102400          SUB 0,0
44 00030' 125112          MOVL# 1,1,SZC      ;CHECK FOR NEGATIVE NUMBER
45 00031' 020424          LDA 0,MINUS
46 00032' 006006$         JSR @.FLOT         ;CONVERT SAMPLE TO FLOATING POINT
47 00033' 030424          LDA 2,ADRBF        ;LAOD THE STORAGE ADDR
48 00034' 041000          STA 0,0,2
49 00035' 045001          STA 1,1,2          ;STORE FLT PT SAMPLE
50 00036' 010421          ISZ ADRBF          ;INC STORAGE ADDR
51 00037' 010420          ISZ ADRBF          ;FOR FLT PT WORDS
52 00040' 014416          DSZ NSAMP          ;DECR THE SAMPLE COUNTER
53 00041' 000402          JMP CONT           ;MORE SAMPLES TO BE TAKEN
54 00042' 000404          JMP DONE           ; FINISHED
55
56 00043' 063614 CONT     SKPDN RTC          ;WAIT FOR CLOCK TO FINISH
57 00044' 000777          JMP .-1
58 00045' 000755          JMP SAM            ;TAKE ANOTHER SAMPLE
59
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
  0002   SUBR2
01 00046' 060231 DONE:    NIOC AIN          ;CLEAR AIN
02 00047' 002007$         JMP @.LINE        ;DONE RETURN TO BASIC
03
04 00050' 006010$ER:      JSR @.ERRA        ;ERROR RETURN TO RTBASIC (BAD EXPR)
05 00051' 000103          103               ;ERROR NUMBER
06
07
08 00052' 126520 SSAMP:   SUBZL 1,1         ;STORE 1 SAMP
09 00053' 044403          STA 1,NSAMP       ;SAMPLE COUNTER
10 00054' 000730          JMP STADR
11
12
13 00055' 177777 MINUS:   177777
14 00056' 000000 NSAMP:   0
15 00057' 000000 ADRBF:   0
16 00060' 177777 ADRSZ:   ARRSZ
17
18
19                .END
```

APPENDIX B

SHORT SUMMARY OF RTBASIC STATEMENTS

	Purpose	Format
1. I/O		
OPN	Open a File on a channel.	OPN "FILEN",3
CLS	Close a channel.	CLS 3
PRG	Read in a program.	PRG "BPROG"
RAND	Read or write an array from a random record file.	RAND CHN 3,A,10,"R"
DOS	Close all Files and return to DOS	DOS
LIST #n	List Program on channel.	LIST #2
PRINT #n PRINT CHN	Print on a channel.	PRINT CHN 2,A1, PRINT #2,A1
INPUT #n INPUT CHN	Input from a channel.	INPUT CHN 2,A2
2. Digital Register		
OGPR	Output an array or integer to GPR and perform control function.	OGPR A, "S" OGPR I, "P"
IGPR	Input an array or integer from GPR and perform control function.	IGPR B, "P"
3. Real Time Clock		
CLK	Set variable equal to the value of RTC; start, stop and set period of clock.	CLK A CLK "S" CLK "D" CLK "S2"
4. Character Generator		
WRITE	Display numbers and characters (identical to PRINT) on selected display device.	WRITE "ABCD1#" WRITE A(I)

	Purpose	Format
DSPLY	Display user programs on selected display device.	DSPLY 200 DSPLY

5. Display and Plotter Equipment

XYSEL	Select devices to be used and intensification mode.	XYSEL "APS"
PLOT	Position beam or pen to X, Y position and plot.	PLOT 1, SIN(X)
YPLOT	Plot array, step along X coordinate	YPLOT A, 3 YPLOT C
BPLOT	Plot X, Y from two arrays	BPLOT X,Y
PEN	Move the X-Y plotter pen up and down.	PEN "U" PEN "D"

6. Digital to Analog Output Channels

ADA BDA	Output simple variable or array and select conversion mode on A or B channel.	ADA X,Y
DAB	Output two integers or two arrays and select conversion mode.	DAB C, D

7. Signal Processing Functions

FFT	Fast Fourier Transform	FFT A, B FFT S, "R"

8. Analog to Digital Converters

SAMP	Sample analog channel	SAMP A,7,1,5,B SAMP A1,Ø

9. Multi-level Interrupt System

ONI	Handle Interrupt	ONI "C" ONI 7 GOSUB 335
WAIT	Wait for sampling or analog output to finish.	SAMP WAIT

10. User Subroutine

CALL	Call user routines (written in NOVA assembly language) with or without arguments.	CALL "SUBR1" CALL "SUBR5" A, 7, "STRING"

APPENDIX C

RTBASIC ERROR MESSAGES

The following error messages, which pertain to the RTBASIC statemetns are an addition to the error messages in Single-User BASIC.

ERRORS

		Applicable Statements
37	CALL - Illegal Subroutine Call	(CALL)
40	Arrays have different dimensions	(FFT,BPLOT)
41	Zero Dimension - Not an array	(RAND FFT BPLOT YPLOT)
48	Buffer size too large	(SAMP)
53	Array Dimension too small	(GPR FFT RAND)
61	No condition set for this interrupt	(ONI)
70	Period (1/Frequency) out of range	(SAMP,ADA,BDA,DAB)

I/O ERRORS

The following error messages will be generated when an error occurs in I/O statement. An interrupt from an unknown device will be cleared.

	MEANING
100	ILLEGAL CHANNEL NUMBER
101	ILLEGAL FILE NAME
103	ILLEGAL COMMAND FOR DEVICE
105	ATTEMPT TO WRITE AN EXISTENT FILE
106	END OF FILE
107	ATTEMPT TO READ A READ PROTECTED FILE
108	ATTEMPT TO WRITE A WRITE PROTECTED FILE
109	ATTEMPT TO CREATE AN EXISTENT FILE
110	ATTEMPT TO REFERENCE A NON-EXISTENT FILE
113	ATTEMPT TO REFERENCE A FILE NOT OPENED.
117	ATTEMPT TO USE A CHANNEL ALREADY IN USE
118	LINE LIMIT EXCEEDED ON READ OR WRITE LINE
120	PARITY ERROR ON READ LINE
122	ATTEMPT TO ALLOCATE MORE MEMORY THAN AVAILABLE
123	OUT OF DISK SPACE
124	FILE READ ERROR
125	UNIT NOT PROPERLY SELECTED
130	ILLEGAL DEVICE NAME

SINGLE-USER BASIC ERROR MESSAGES

00 Format Error
01 Illegal Character
02 Syntax Error
04 System Error
05 Illegal Statement Number
06 Too Many Variable Names
07 Spelling Error
08 Spelling Error
09 No Such Word
10 Incorrect Subscript Closure
11 Incorrect Parenthesis Closure
12 Not a Keyboard Command
13 No Such Line Number
14 Storage Overflow (While Inputting Program)
15 Read Statement is Out of Data
16 Arithmetic Overflow (Number Too Large)
17 No END or STOP Statement in Program
18 Too Many Nested GOSUBs
19 Too Many RETURNs
20 Too Many Nested FORs
21 FOR Without NEXT
22 NEXT Without FOR
23 Out of Storage (While Assigning Variable Storage)
24 Array Too Large
25 Attempt to Dimension Simple Variable
26 Variable Name is Not Dimensionable
28 Redimensioned Array is Larger Than Previously Defined
29 Expression is Too Complex
30 Illegal Format in Defined Function
31 Subscript Exceeds Dimension
32 Undefined User Function
33 Too Many Nested Functions
34 Negative Subscript
35 Function Not Yet Implemented

APPENDIX D

RTBASIC KEYBOARD COMMANDS

The following is a list of RTBASIC statements which can be executed as keyboard commands. If a statement cannot be executed in keyboard mode, the appropriate error will be returned.

OPN
CLS
XYSEL
DSPLY
PRG
DOS

APPENDIX E

PROGRAMMING TIPS FOR RTBASIC

1. It is sometimes necessary to sample or output more than 1024 points which is the largest size array allowed. It is possible to output or sample continously more than 1024 points in the following manner:

```
10 DIM A (3,255)
20 DIM B (3,255)

30 SAMP A,5,10000/F
31 SAMP B,5,10000/F
40 WAIT
```

The two SAMP statements will sample 2048 points with almost no delay between points

2. If a CLK statement has been used and is no longer needed a

```
CLK "D"
```

statement should be used to stop the clock to save processing time which would be used to continue servicing clock interrupts.

3. An easy way to read in parameters in RTBASIC is to use the front pannel knob and a simple sample statement

```
SAMP P1,Ø
```

4. The trigger switches are very useful in controlling Program execution. The following short program allows a signal to be sampled and displayed continuously and then resampled when trigger switch Ø is pressed.

```
 5  Let F = 100
10  ONI Ø
11  GOTO 30
20  XYSEL "ABS"
25  DIM A(3,255)
30  SAMP A,4,1000/F
40  YPLOT A
50  GOTO 40
```

STEREO DISPLAY OF VECTORCARDIOGRAMS

by

Jerome Ney Sheridan

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE, 1971

Signature of Author.....Jerome N. Sheridan.....
Department of Electrical Engineering

Certified by.....S.K. Burns.....

.....Roger Mark.....
Thesis Supervisors

Accepted by.....Robert L. Kyhl.....
Chairman, Departmental Undergraduate Thesis Committee

PRECEDING PAGE BLANK NOT FILMED

STEREO DISPLAY OF VECTORCARDIOGRAMS

Submitted to the Department of Electrical Engineering on May fourteenth 1971 in partial fulfillment of the requirements for the degree of Bachelor of Science.

ABSTRACT

A three-dimensional display of the vectorcardiogram loop showing the locus of the tip of the cardiac dipole vector was obtained through the use of a computer to give a stereoscopic projection of points on the loop. Features of the computer program are discussed and some typical results are presented.

Thesis Supervisors:

Stephen K. Burns, Associate Professor of Electrical Engineering

Roger G. Mark, Assistant Professor of Electrical Engineering

ACKNOWLEDGEMENTS

The author is indebted to Professor Stephen K. Burns and Professor Roger G. Mark for their guidance and patience.

Appreciation is also extended to J. B. Walters for his timely suggestions which saved the author many hours of futile effort.

The research for this thesis was done using the facilities of the Research Laboratory of Electronics.

Table of Contents

Introduction		p. 4
Vectorcardiography		p. 4
Display Systems		p. 7
The Display Program (VCGD)		p. 9
Conclusions		p. 16
Appendix A:	Rotation and Projection of Coordinate Systems	p. 17
Appendix B:	Use of Display Program (VCGD)	p. 23
Appendix C:	Basic Flow Charts for the Display Program	p. 29
Appendix D:	Computation of Areas	p. 34
References		p. 37

STEREO DISPLAY OF VECTORCARDIOGRAMS

Introduction

The vectorcardiogram records the spatial orientation and magnitude of the cardiac dipole as it varies throughout the cardiac cycle, and is known to contain clinically useful information about the physiological status of the heart muscle. To date, however, the clinical usefulness of vectorcardiography has been limited. One limiting factor has been the lack of an adequate display system. The aim of this thesis project was to explore ways in which a computer could be used to fulfill this need for a better display system. The result is a computer program which computes a stereo projection of the vectorcardiogram and displays it on an XY display device. Other parameters of the displayed signal are also computed and displayed.

Vectorcardiography

The depolarization of the muscle of the heart which prefaces contraction is not an instantaneous process. It is initiated at one point and spreads throughout the myocardium. The resulting discontinuity between polarized and depolarized muscle produces an electric field which in turn creates potential differences between different points of the body surface.

Modern scalar electrocardiography is based on the following hypotheses concerning this process:

(1) the heart acts as a dipole

(2) this dipole is located centrally in the chest and does not move during the cardiac cycle

(3) the standard electrode positions (figure 1) are equidistant (electrically) from the dipole

(4) the human torso can be assumed to be shaped like a sphere.

Of these, the dipole hypothesis retains universal acceptance. It has been shown by Frank and others, however, that (2), (3), and (4) are unwarranted.[1] They are all still used, though, since they seem to give adequate results in conventional electrocardiography, and are definite aids to intuition.

Vectorcardiography follows directly from the dipole hypothesis. The varying voltages recorded at the body surface imply that the dipole is varying in magnitude and/or direction with time. If a set of "orthogonal" voltages could be derived from the body surface potentials, then the path of the moving vector could be determined. If all four of the above-mentioned hypotheses were valid, then a set of orthogonal voltages could be derived from any four nonplanar points on the surface of the body. Instead it has been necessary to resort to empirical determination of the body's true electrical properties. This has led to the development of the so-called corrected lead systems.

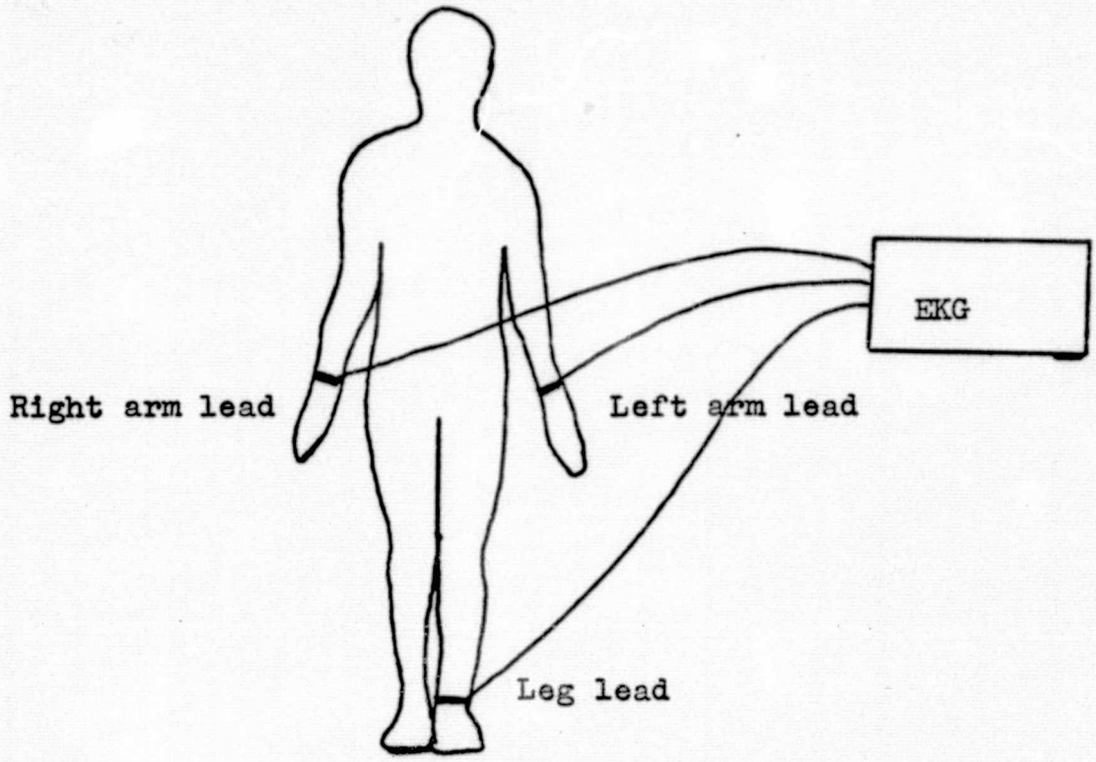


Figure 1. Standard leads

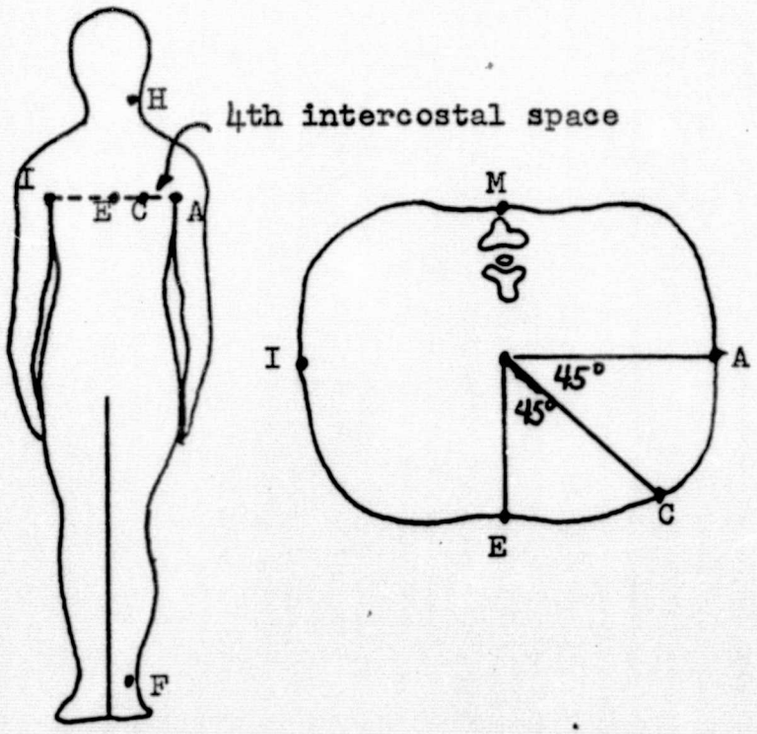


Figure 2. Frank orthogonal lead system. Electrode placement.

The Frank System is one such system which has gained widespread acceptance (figure 2)[2]. A resistive network is used to combine the signals from the seven electrodes in such a way as to result in orthogonality. The three orthogonal leads correspond to coordinate axes oriented as shown in figure 3. The display program to be described was written assuming a Frank System input.

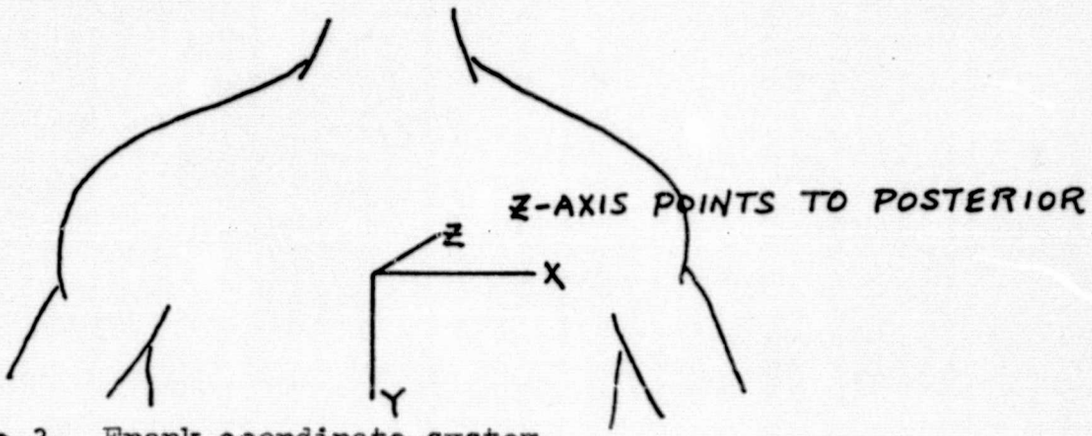


Figure 3. Frank coordinate system.

To summarize, the vector cardiogram records the varying potentials at the surface of the body as a vector quantity. The locus of the head of this vector quantity describes a loop in space and is referred as the vector loop or as simply the loop.

Display Systems

Standard representations of the vector loop are obtained by photographing an oscilloscope display representing each of the orthogonal planes (the XY, XZ, and YZ planes). Most conventional display systems require visualization of the loop from three distinct heartbeats

that were recorded in this manner. Minor variations in the loop from beat to beat make the already difficult visualization process arduous indeed. Also, the bunching of points in the early part of the QRS loop adds to the difficulty, and it is this early electrical activity which is frequently of greatest interest in the diagnosis of myocardial infarctions. Finally, the spatial angles and magnitudes of the instantaneous vectors are not readily obtained from most current display systems.

Workers in the field have made numerous attempts to remedy this situation without much success. Stereoscopic display techniques were developed in the early 1950's, but the stereo pictures produced were poor and the systems cumbersome.[3,4] Computerized methods[5,6] have been developed which utilize various cartographic schemes for representing the three-dimensional loops. However, the expense of the equipment used will probably prohibit the use of these methods clinically. A method for recording vectorcardiograms using the principles of holography is an interesting recent development[7]. It remains to be seen whether a practical system utilizing this technique can be developed.

The display system which is the subject of this thesis utilizes a combination of computer and stereoscopic techniques to achieve a three-dimensional display of the vector loop. This display, which effectively allows

the vector loop to be observed from any angle, also utilizes the power of the computer to obtain parameters of the signal considered relevant to the diagnosis of heart disease.

The display program was implemented on a computer system utilizing a NOVA (Data Generel Corporation, Southboro, Massachusetts) as the basic processor. Peripheral hardware includes an XY plotter, two CRT's, a teletype, a high speed reader/punch, and a 512K word disk. Also included is an analog interface consisting of a sixteen channel analog-to-digital (A/D) converter, four digital-to-analog (D/A) converters, and a character generator. Of these devices, the display program directly utilizes the analog interface and the XY display devices.

The Display Program VCGD

The display program can be thought of as operating in three distinct modes:

(1) the Sample Mode

(2) the Scalar Display Mode

(3) the Stereo Display Mode.

As explained in Appendix B, the different modes are selected by the switches on the front panel of the NOVA.

In the Sample Mode, the vectorcardiograph leads are sampled at one thousand times per second (1 KHZ) and the samples are stored in a buffer. There is a simple level trigger which is used to signal the program when significant

data has been acquired. A delay following the triggering event was also incorporated so that input following triggering could also be captured. The initial internal setting of the trigger level and delay time are adequate for most purposes, but it is possible to vary these parameters while the program is in operation (see Appendix B).

The Scalar Display Mode was incorporated to allow correlation of features of the stereo display to the more familiar scalar leads. Figure 4 shows typical scalar leads of a normal vectorcardiogram. The small vertical markers on each lead define the portion of the signal for which a stereo display will be computed. These markers constitute what is referred to as the "cursor"; the position of the cursor is varied via external controls (A/D converter Channels 2 and 3).

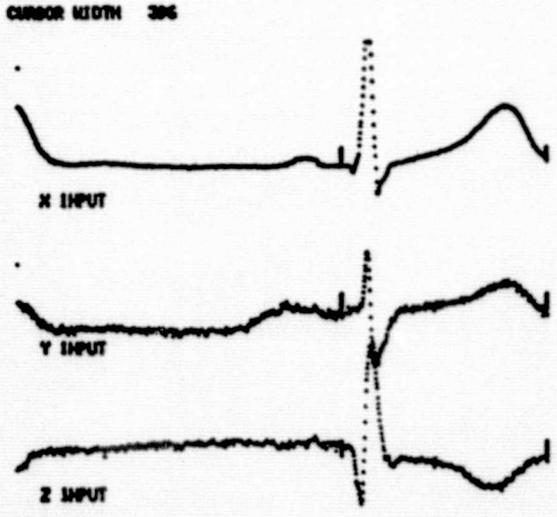


Figure 4. Scalar Display Mode.

In the Stereo Display Mode a stereoscopic image of the portion of the signal selected by the cursor is produced. Reference axes are also displayed for convenience in inter-

preting the vectorcardiogram. (Figure 5 is the stereo projection of the scalar leads seen in figure 4.)
The apparent orientation of the loop can be varied externally via Channels Ø and 1 of the A/D converter. Varying the voltage on Channel Ø amounts to an apparent elevation above (or declination below) the XZ plane. Varying the input to Channel 1 causes apparent rotation about the Y axis. These transformations are achieved by the use of equations which cause a rotational transformation of the coordinate system, followed by a plane projection into two dimensions.

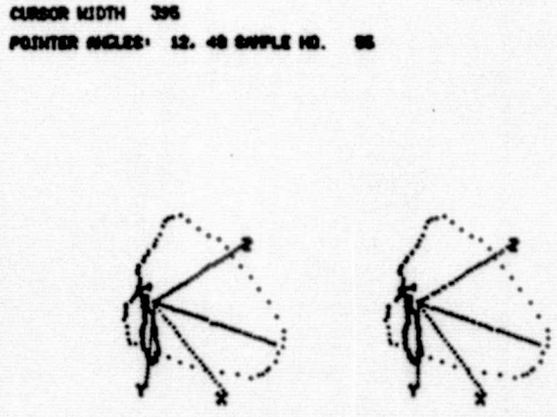


Figure 5. Stereo Display Mode. Stereo projection of scalar leads shown in figure 4.

The stereo effect is achieved by displaying two images of slightly different orientation. The difference between the two images is an offset rotation about the vertical axis(see figure 6).

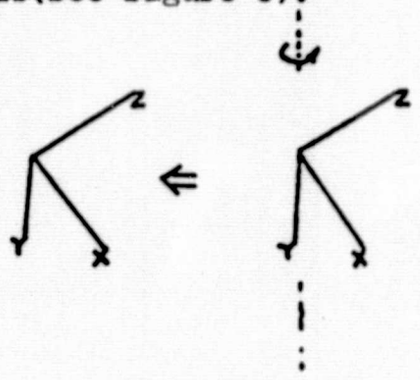


Figure 6. Generation of the stereo effect.

The equations governing the transformation of a point with coordinates (x,y,z) into coordinates (x_L,y_L) and (x_R,y_R) for display are given below:

$$x_R = (\cos\theta)x + (-\sin\theta)z$$

$$x_L = (\cos\theta\cos\beta - \sin\theta\cos\phi\sin\beta)x + (-\sin\phi\sin\beta)y + (-\cos\theta\cos\phi\sin\beta - \sin\theta\cos\beta)z$$

$$y_L = y_R = (\sin\theta\sin\phi)x + (-\cos\phi)y + (\cos\theta\sin\phi)z$$

where θ = rotation angle about Yaxis

ϕ = elevation (declination) angle above (below) the XZ plane

β = offset angle ($\approx 5^\circ$) used to generate the stereo effect.

Appendix A contains a discussion of coordinate transformations and a derivation of the above equations.

ORIGINAL PAGE IS
OF POOR QUALITY

Other features available in the Stereo Display Mode include:

1) A routine which calculates the area of the left stereo projection loop. Appendix D discusses the method used to calculate this area.
2) A routine which generates a stereo pointer to an individual data point and computes the spatial angles of the point. The position of the pointer is externally variable via Channel 7 of the A/D converter.
3) A routine which finds the spatial angles of the mean vector of the displayed points.
4) A feature which allows the three standard views (XY, XZ, and YZ planes) to be readily displayed.

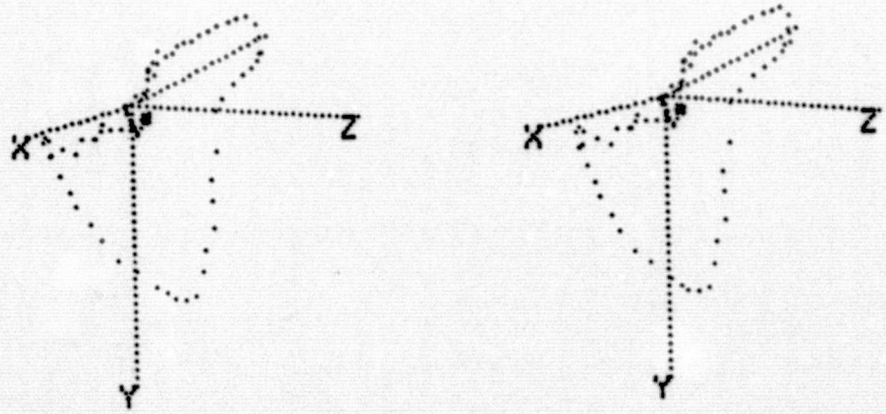
Figure 7 demonstrates the output obtained when features (1), (2), and (3) are selected. "CURSOR WIDTH" refers to the number of samples currently being displayed. "SAMPLE NO." refers to the number of the sample relative to the beginning of the cursor. The meanings of the other captions should be obvious.

Figure 8 demonstrates the output obtainable through the use of feature (4). The stereo views shown are all of the same loop.

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 7. QRS loop with area, mean vector, and pointer features selected.

CURSOR WIDTH 108
POINTER ANGLES: 231, 32 SAMPLE NO. 65
MEAN VECTOR: 46, 122
AREA: 23312



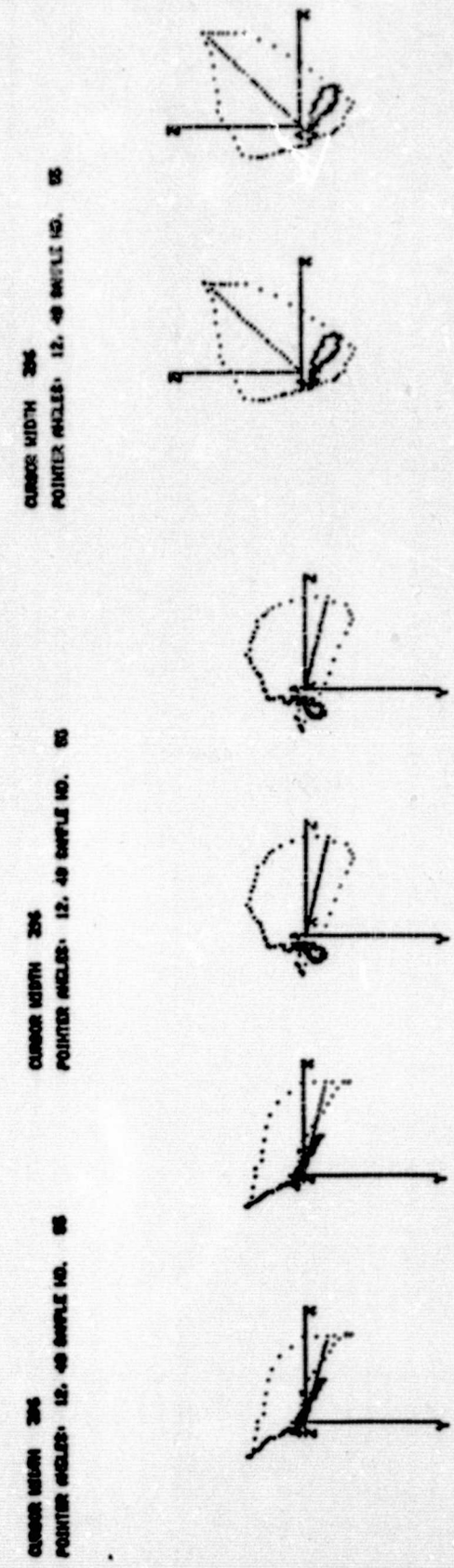


Figure 8. Three standard views of the same loop selected via panel switches (see Appendix B). Small loops, such as these, readily enable stereo viewing by crossing the eyes while staring at a midpoint between the two images.

Conclusions

The display program does a good job of displaying vector loops. Compared to examples of loops found by conventional means, it does an excellent job. The requirement of special equipment for viewing the stereo image is a handicap, but previous versions of the program, which did not have stereo capability, demonstrated the difficulties encountered in working with the plane projection alone. At this point a clinical evaluation of the system would be desirable in order to determine what direction (if any) future development should take.

Whether this system or similar systems will ever be used clinically is open to question. The big question is one of economics. In spite of the relatively low cost of the equipment used, it would cost at least $15,000, perhaps more, to build a computer-based vector cardiograph from scratch. The price of hardware keeps dropping, but such a project will probably remain a fairly expensive proposition for some time to come. Another possibility is the development of special hardware to do the job. A recirculating memory could be used to hold the signal and the transformation function could be implemented using nonlinear analog devices which are currently becoming available. The resulting display would be less versatile but probably a good deal less costly.

APPENDIX A: ROTATION AND PROJECTION OF CO-ORDINATE SYSTEMS

Introduction

A three dimensional event is to be displayed on an XY display device. Equations of rotation and projection must therefore be developed which preserve the apparent spatial properties of the signal while reducing it to two dimensions.

Method of Projection

A non-perspective plane projection was chosen in the interest of speed and ease of implementation. The more commonly used perspective method[8] was rejected because:

(1) a great deal more computation time would be required

(2) it was felt that in this case, perspective would convey little or no depth information to the observer.

Figure A shows an example of a plane projection onto

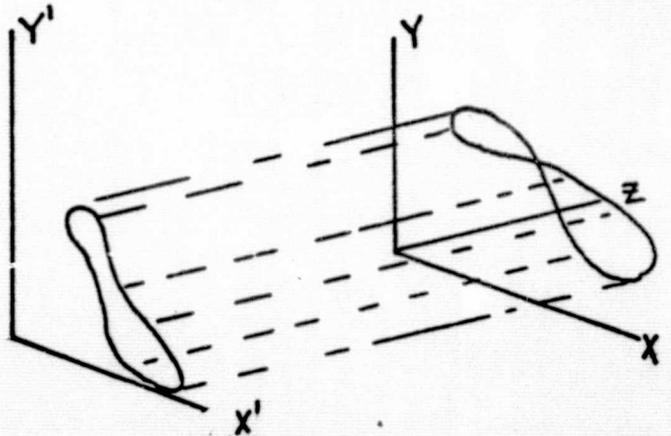


Figure A.

the X'Y' plane from an XYZ coordinate system. (The X'Y' plane is parallel to the XY plane.)

It can readily be seen from the figure that for each point (x_n, y_n, z_n) the projected point is just (x_n, y_n). In other words, this plane projection amounts to throwing away z_n.

Now suppose there is a collection of points $\{x_n, y_n, z_n\}$. As in figure A, one can find a projection of these points onto the XY plane by simply throwing away the z_n. The more general question, however, is the following: how does one find the plane projection onto an arbitrary plane?

The easiest solution to this problem is to consider how the set $\{x_n, y_n, z_n\}$ can be transformed into a new collection of points $\{x'_n, y'_n, z'_n\}$. The desired projection is then $\{x'_n, y'_n\}$. This transformation amounts to a rotation of coordinate systems.

Rotation of Coordinate Systems

The general equations for rotating an XYZ coordinate system can be found in any mathematics reference book[9] and are as follows:

$$x' = a_1x + b_1y + c_1z \qquad (1)$$

$$y' = a_2x + b_2y + c_2z \qquad (2)$$

$$z' = a_3x + b_3y + c_3z \qquad (3)$$

The a's, b's, and c's represent direction cosines corresponding to the cosines of the angles between the new axes and the old.

The above tranformation, while perfectly valid, presents

problems in that the angles which yield the a's, b's, and c's are inter-dependent. An independent set of three angles could be selected, but the remaining problem of selecting a desired projection plane in terms of these angles is not easy.

To get around this problem, the equations for a transformation of slightly less generality were developed. The loss of generality is minor; projections into any arbitrary plane can still be obtained, the only difference being that in general, the projection cannot itself be rotated. This transformation will be referred to as a polar transformation since the transformation parameters correspond to the angles of variation in a polar coordinate system.

The polar transformation was developed by successive application of the familiar equations for transformation of a point (u,v) in a UV coordinate system which is rotated by an angle α:

$$u' = u \cos\alpha + v \sin\alpha \qquad (4)$$

$$v' = -u \sin\alpha + v \cos\alpha \qquad (5)$$

where u',v' are the coordinates or the point in the new system (figure B).

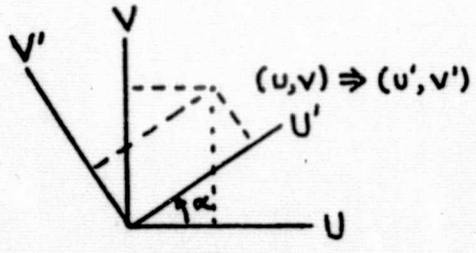


Figure B.

The polar transformation can be thought of as a sequence of transformations. The first transformation is a rotation about the Y axis (figure C).

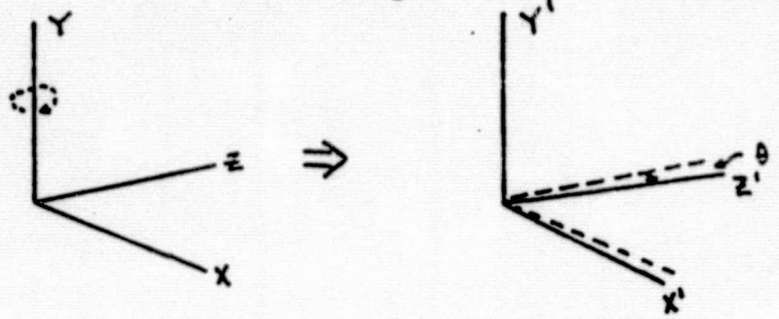


Figure C. Rotation about the Y axis.

Apply equations (4) and (5):

$$x' = x\cos\theta - z\sin\theta \tag{6}$$

$$y' = y \tag{7}$$

$$z' = x\sin\theta + z\cos\theta \tag{8}$$

The second transformation corresponds to a rotation about the X' axis (figure D).

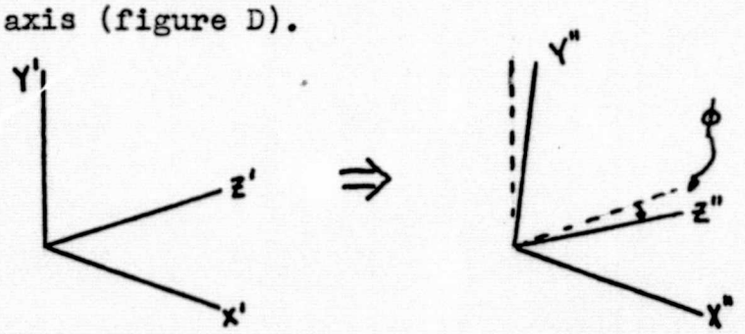


Figure D. Rotation about the X' axis.

The resulting equations are:

$$x'' = x' \tag{9}$$

$$y'' = y'\cos\phi + z'\sin\phi \tag{10}$$

$$z'' = \ldots \sin\phi + z'\cos\phi \tag{11}$$

Substituting for x', y', and z' from equations (6), (7), and (8):

$$x'' = x\cos\theta - z\sin\theta \tag{12}$$

$$y'' = y\cos\phi + (z\cos\theta + x\sin\theta)\sin\phi \tag{13}$$

$$z'' = -y\sin\phi + (z\cos\theta + x\sin\theta)\cos\phi \tag{14}$$

The equations for x" and y" represent the desired polar transformation with plane projection.

Stereo Projection

To achieve a stereo effect two projections of slightly different orientation are desired. Equations (12) and (13) suffice for the right image of the stereo pair. The equations for the left image are obtained by rotating the X"Y"Z" system about the Y" axis (figure E). The angle of rotation, β, is fixed.

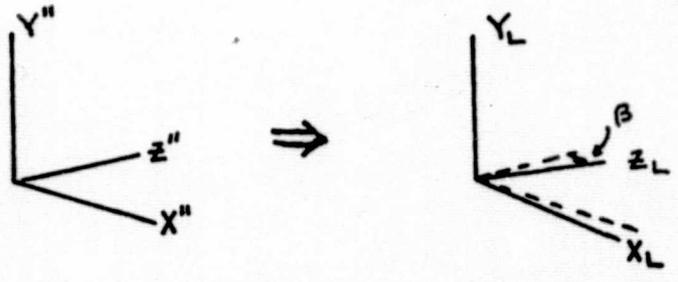


Figure E. Rotation about the Y" axis.

Again applying equations (4) and (5):

$$x_L = x''\cos\beta - z''\sin\beta \tag{15}$$

$$y_L = y'' \tag{16}$$

$$z_L = x''\sin\beta + z''\cos\beta \tag{17}$$

Substituting from (12), (13), and (14):

$$x_L = (x\cos\theta - z\sin\theta)\cos\beta - (-y\sin\phi + (z\cos\theta + x\sin\theta)\cos\phi)\sin\beta \quad (18)$$

$$y_L = y\cos\phi + (z\cos\theta + x\sin\theta)\sin\phi \quad (19)$$

$$z_L = (x\cos\theta - z\sin\theta)\sin\phi + (-y\sin\phi + (z\cos\theta + x\sin\theta)\cos\phi)\cos\beta \quad (20)$$

In the Frank coordinate system the Y axis corresponds to -Y in the above derivation. This is fixed by substituting (-y) for y in the final equations. Below are the transformation equations as actually implemented:

$$x_R = (\cos\theta)x + (-\sin\theta)z \quad (21)$$

$$x_L = (\cos\theta\cos\beta - \sin\theta\cos\phi\sin\beta)x + (-\sin\phi\sin\beta)y + (-\cos\theta\cos\phi\sin\beta - \sin\theta\cos\beta)z \quad (22)$$

$$y_L = y_R = (\sin\theta\sin\phi)x + (-\cos\phi)y + (\cos\theta\sin\phi)z \quad (23)$$

APPENDIX B: USE OF DISPLAY PROGRAM (VCGD)

Basic Operating Instructions

INPUT

All inputs must be restricted to 2 V maximum to avoid damaging the A/D converter.

Channels Ø and 1 are interpreted as angles of rotation and elevation respectively (see figure 3).

Channels 2 and 3 define cursor parameters position and length respectively.

Channels 4,5, and 6 are respectively the X,Y, and Z leads from a Frank System vectorcardiograph. (Input from other systems could possibly be used by changing the system axis definitions through the use of inverters.)

Channel 7 defines the pointer position.

OUTPUT

XY Display. (CRT is the normal output mode. The XY plotter can also be used; see SWITCH 9 under Program Features.)

LOADING

Load using binary loader.

START/RESTART

Switches set at ØØØØØ3.

Program Features

The program has a number of features which are selected by the switches on the front panel of the NOVA. The following list contains a description of the function of each of the switches.

SWITCH	SWITCH POSITION Up = 1 Down = Ø	FUNCTION
Ø	1	Selects sample mode. Input at Channels 4, 5, and 6 is sampled at 1 KHZ and stored in sample buffer.
	Ø	Exit from sample mode is enabled. Exit will occur at completion of next triggering event (see figure 1).
1	1	Stereo display mode is selected (see figure 3).
	Ø	Scalar display mode is selected (see figure 2).
2	1	Locks display at current setting of orientation controls (channels Ø and 1).
3	1	Selects XY standard view.
4	1	Selects YZ standard view.
5	1	Selects XZ standard view.
6	1	Area of the left stereo projection is computed and displayed.
7	1	Computes and displays angles of mean vector of the display.
8	1	A stereo pointer to the data point selected via channel 7 is displayed as well as its angular orientation and sample number.
9	1	Use of the plotter is enabled if

SWITCH	SWITCH POSITION (Up = 1, Down = Ø)	FUNCTION
9, cont'd		SWITCH Ø = Ø (i.e. if the program is in either the scalar display mode or the stereo display mode). When SWITCH 9 is set to 1, the machine will halt. At this time the plotter should be turned on and set up for plotting. After the CONTINUE switch is pressed the computer will execute the plot and again come to a halt. At this time plotting can either be continued by pressing CONTINUE or discontinued. If it is desired to discontinue plotting: first set SWITCH 9 to Ø and turn the plotter off; them press CONTINUE and the program will resume normal operation.
10,11		Not used.
12	1	Allows the manual setting of the trigger level and delay parameters. Operative only when program is in the sample mode (SWITCH Ø = 1).
13,14,15		Not used.

ORIGINAL PAGE IS
OF POOR QUALITY

SAMPLLING MODE (SWØ=1)

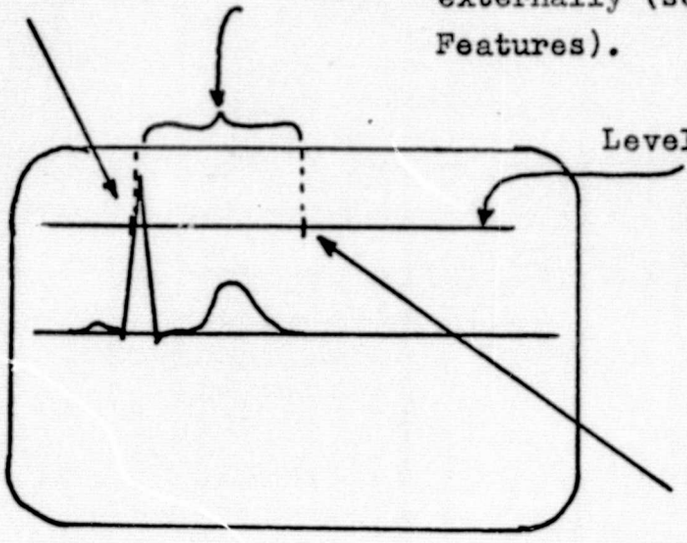


Figure 1. Sample Mode. SWØ=1. (However if SwØ=Ø program will remain in the Sample Mode until a triggering event plus a delay occurs.)

ORIGINAL PAGE IS OF POOR QUALITY

Scalar Display Mode (SW1=Ø,SWØ=Ø)

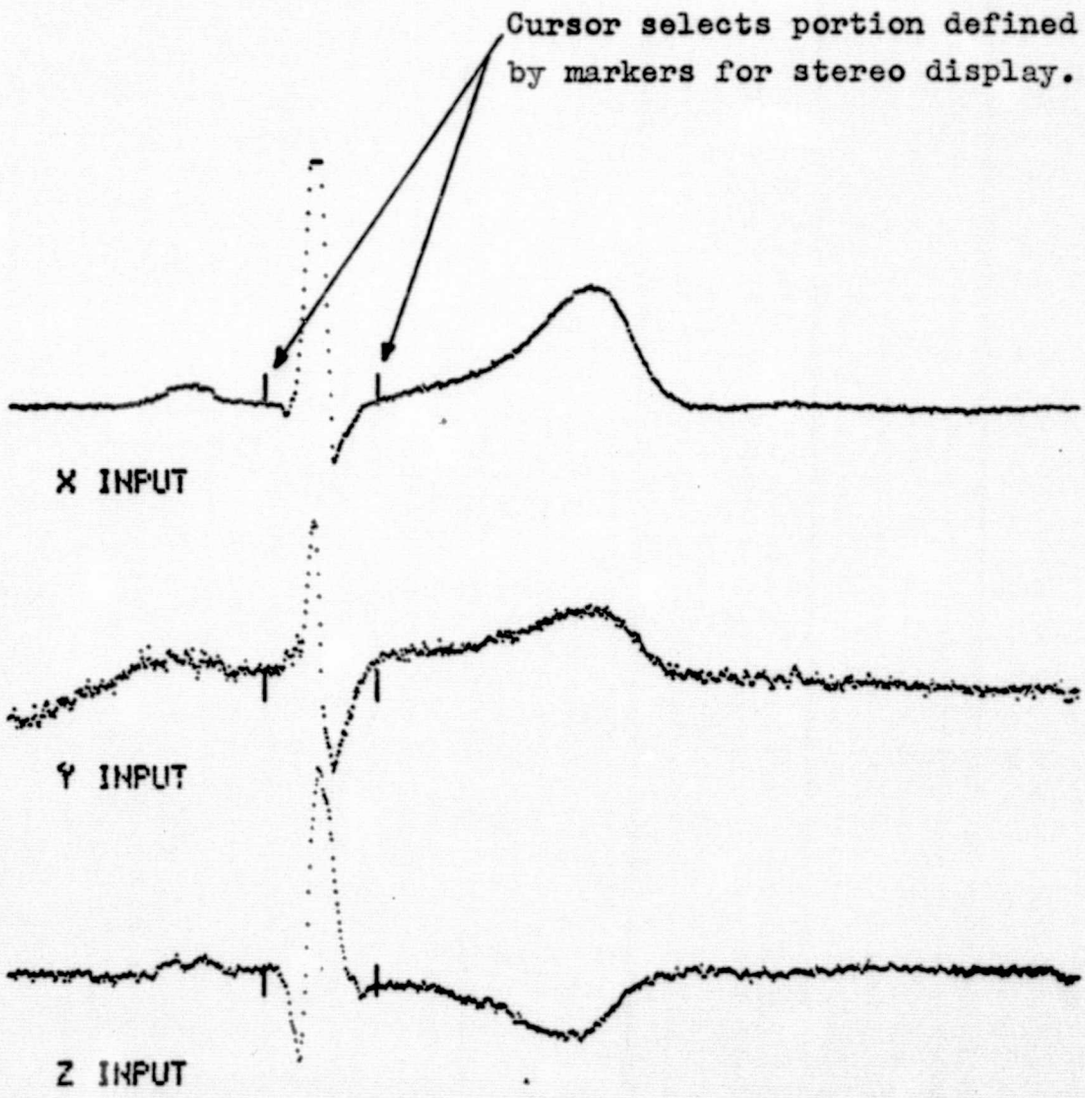
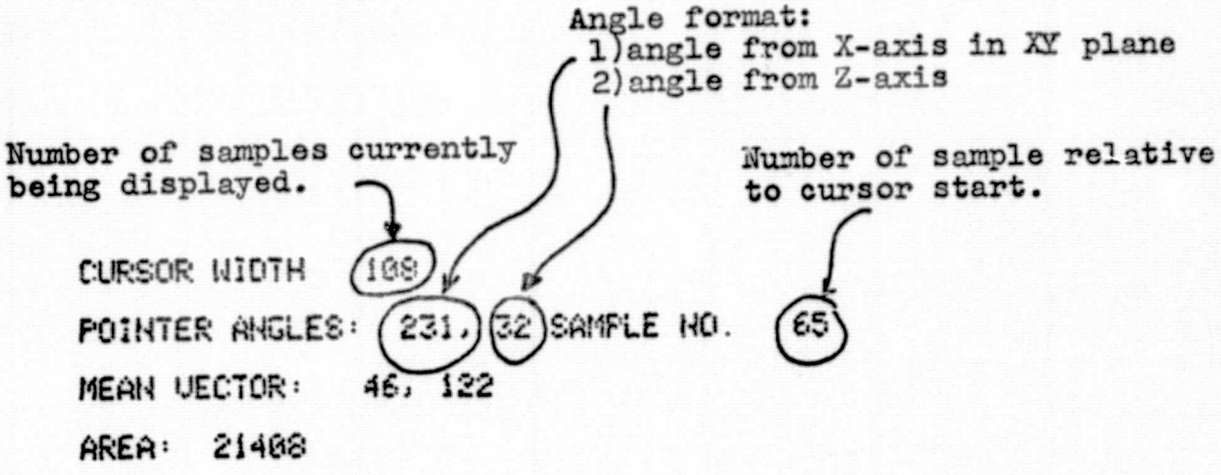
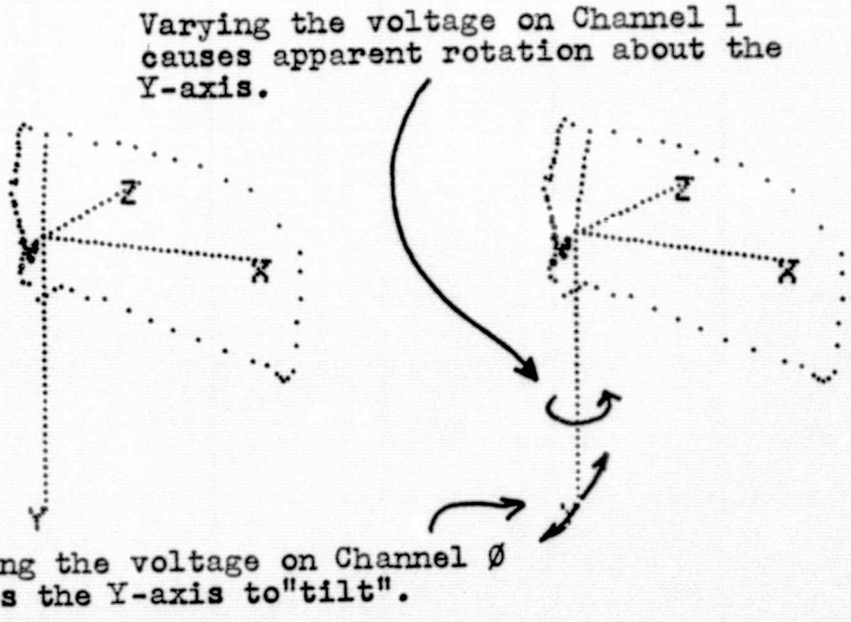


Figure 2. Scalar Display Mode.

Stereo Display Mode (SW1=1,SWØ=Ø)





ORIGINAL PAGE IS
OF POOR QUALITY

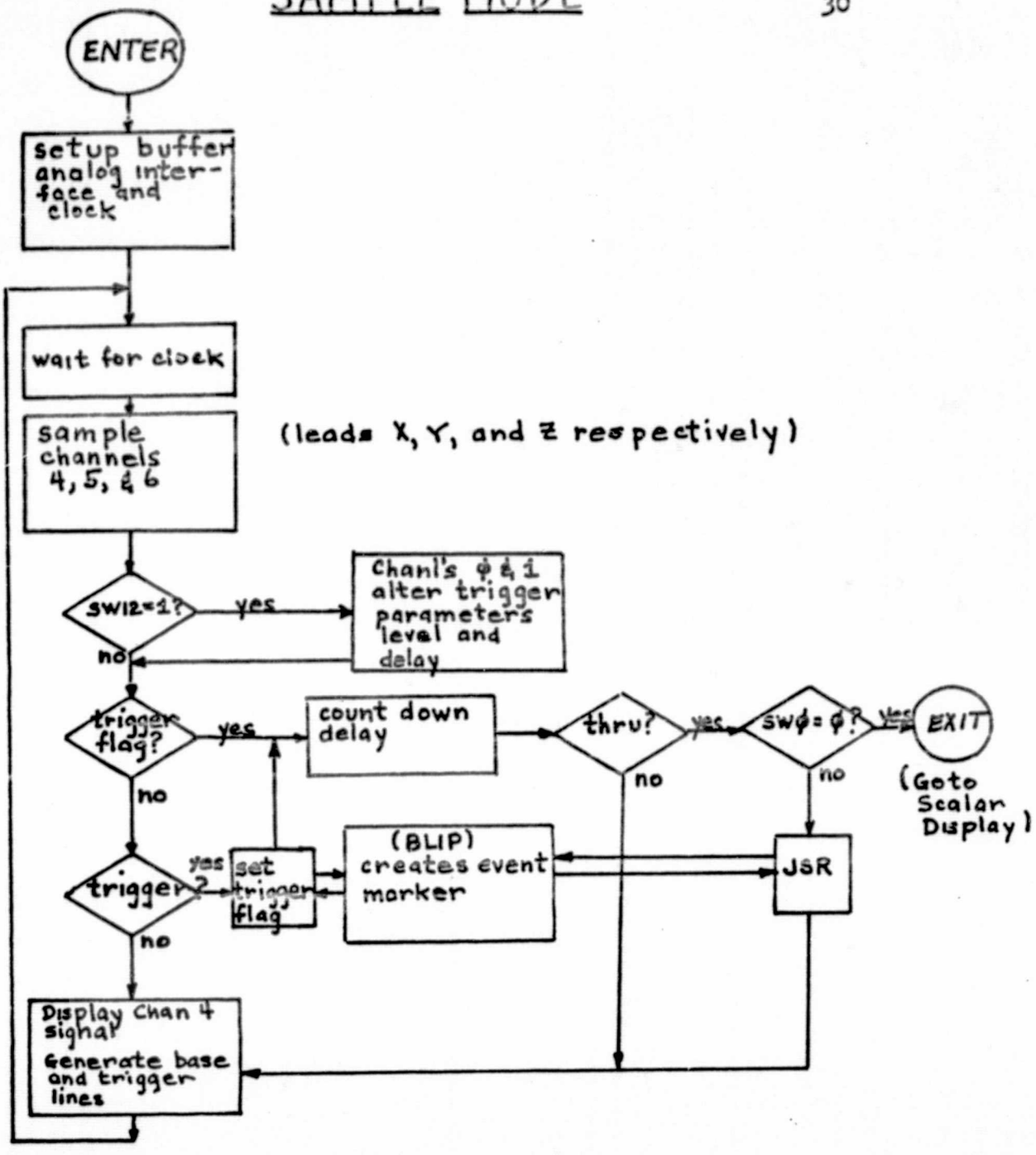
Figure 3. Stereo Display Mode.

APPENDIX C

BASIC FLOW CHARTS FOR THE DISPLAY PROGRAM (VCGD)

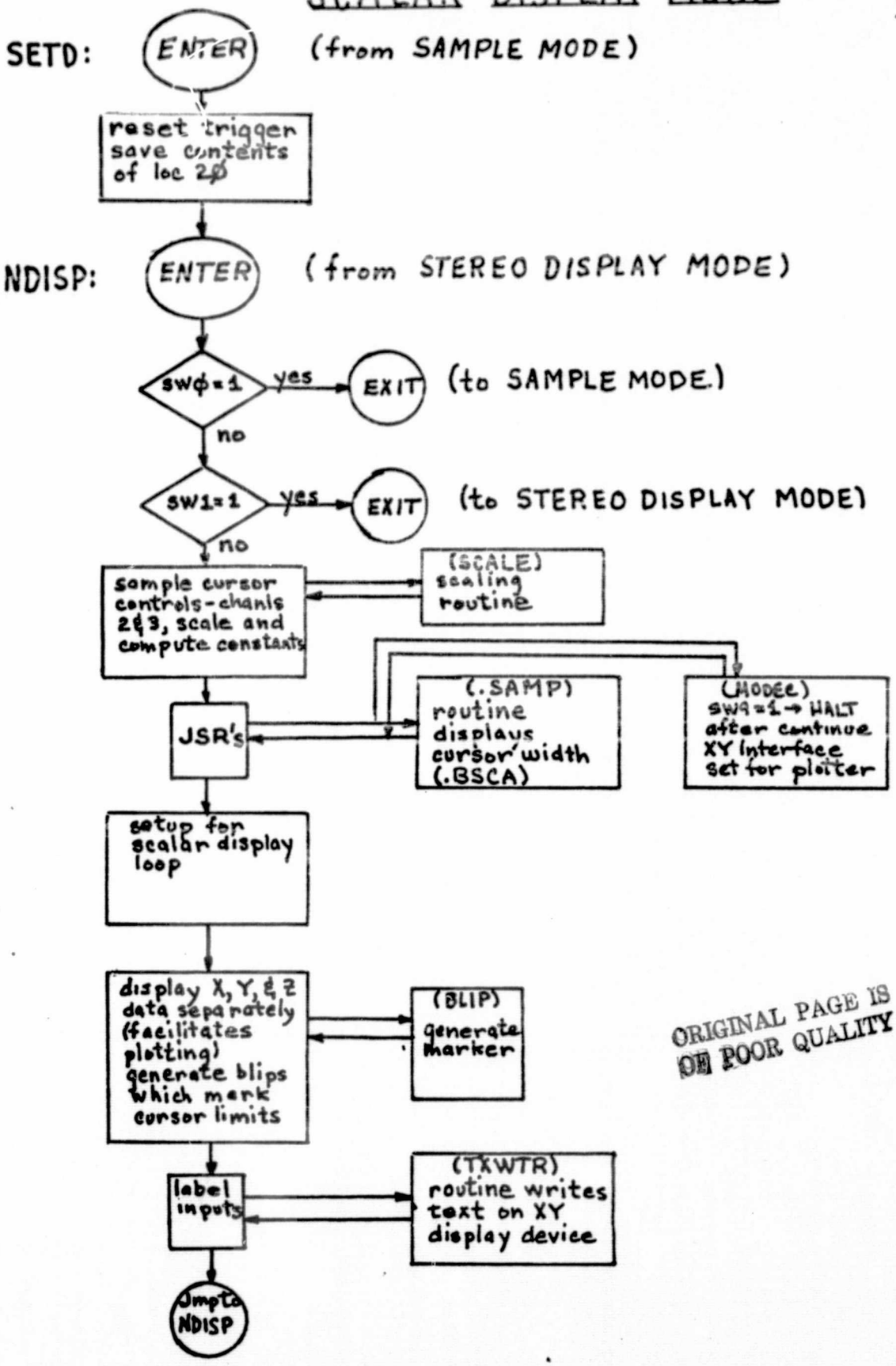
SAMPLE MODE

.RTDI:



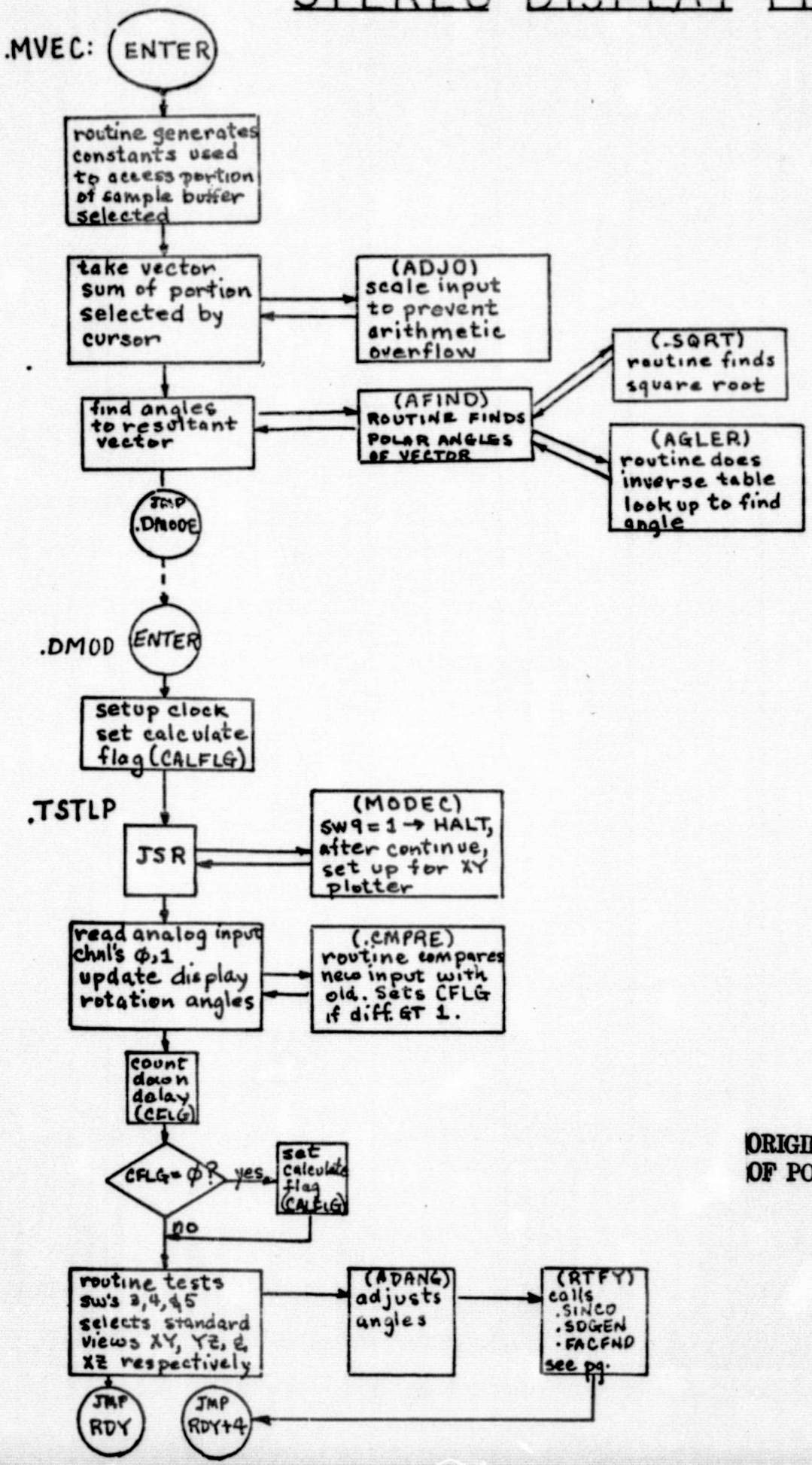
ORIGINAL PAGE IS
OF POOR QUALITY

SCALAR DISPLAY MODE

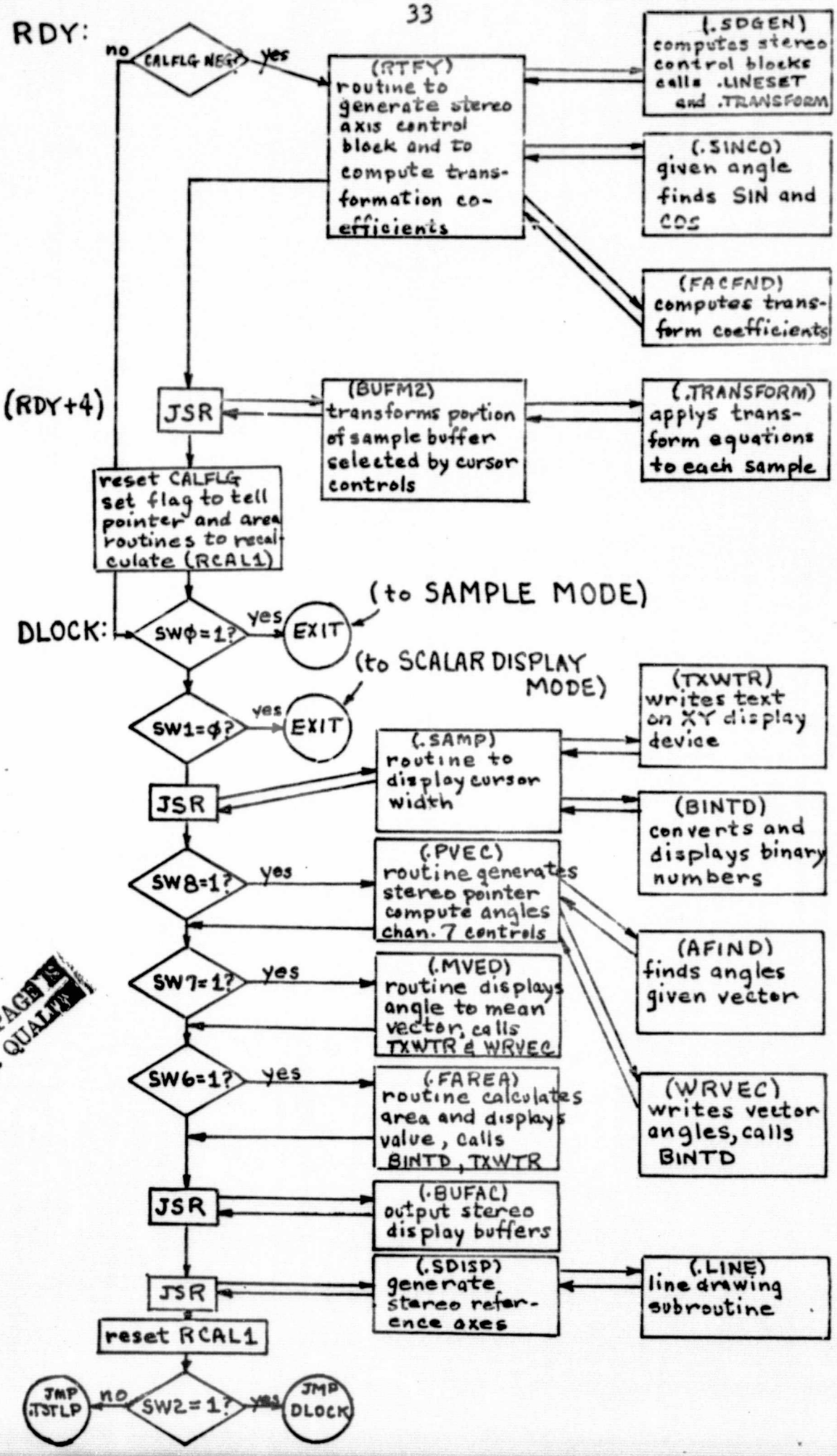


ORIGINAL PAGE IS
OF POOR QUALITY

STEREO DISPLAY MODE



ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS OF POOR QUALITY

APPENDIX D

Computation of Areas

The problem is one of finding the area enclosed (figure 1) by a polygon described by a sequence of coordinates $(x_o y_o, x_1 y_1, \ldots, x_n y_n)$. Since the y-coordinates are known in advance to be restricted to the range $-512 \leq y \leq 512$, the addition of 512 to each y-coordinate will result in all of the y-coordinates being positive:

$$(x_o y_o, x_1 y_1, \ldots, x_n y_n) \rightarrow (x_o y_o + 512, x_1 y_1 + 512, \ldots, x_n y_n + 512) \quad (1)$$

This will amount to a shift upward in the XY plane, but the area enclosed by the loop will remain the same (figure 2).

Referring to figure 2, it can be seen that the areas of the shaded regions are respectively:

$$\text{Area A} = (x_1 - x_o)\frac{(y_o + y_1)}{2} \quad (2)$$

$$\text{Area B} = (x_5 - x_4)\frac{(y_4 + y_5)}{2} \quad (3)$$

Since $x_1 > x_o$, equation (2) will be a positive quantity while equation (3) will obviously be negative since $x_5 < x_4$. Proceeding around the loop in this fashion, summing the increments will result in the area of the loop since the negative increments will partially cancel out the positive increments (figure 3). Clearly if the loop were to be transversed in an opposite fashion one would end up with a negative area. The result would be the

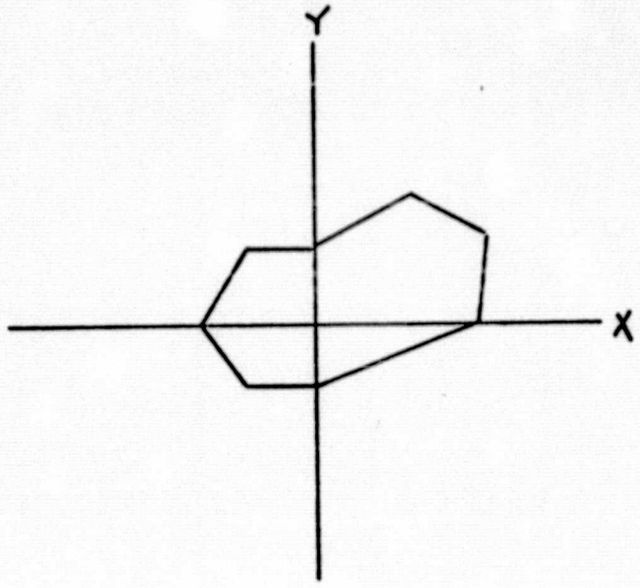


Figure 1. Polygonal area to be computed.

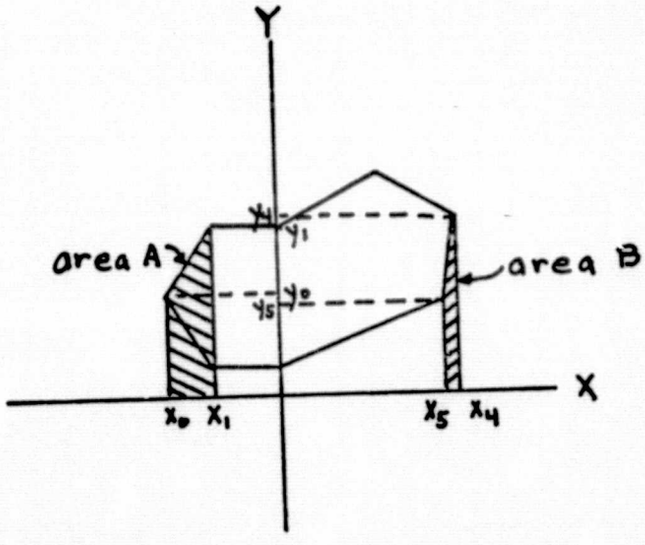


Figure 2. Algorithm operating on shifted polygon.

negative of the correct area.

The general result is as follows:

$$\text{Area} = \left| (x_0 - x_n)\frac{(y_0 + y_n)}{2} + \sum_{o}^{n} \left[(x_{n+1} - x_n)\frac{(y_n + y_{n+1})}{2} \right] \right|$$

this term "closes" the loop

The result, however, is good only for simple loops, i.e. loops in which a boundary never crosses another boundary. For example, the area of a perfect figure eight will be zero when calculated using the above method.

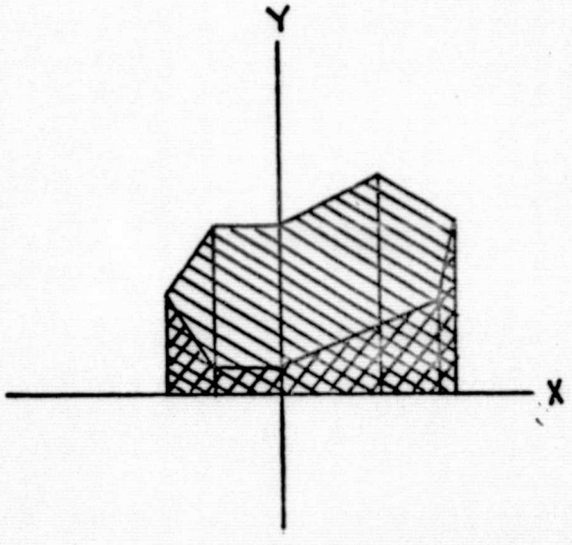


Figure 3. Algorithm yields the single-crosshatched area as area of loop.

References

(1) Frank, E., General theory of heart vector projection. Circulation Res. 2:258, 1954

(2) Frank, E., The image surface of a homogeneous torso.

(3) Guyton, A. C., Crowell, J. W., A stereovectorcardiograph, J. Lab. and Clin. Med., Vol. 40, p. 726, 1952.

(4) Kwoczynski, J. D. Keller, J. J., and Eskiel, J. S., The Stereoscopic Method and Oscilloscopic Arrangement for Spatial Vectorcardiography, Cardiology, Vol. 24, p. 276, 1954.

(5) Brinberg, L., A New Display in Vectorcardiography, Am. Heart J., Vol. 74, p. 855, 1967.

(6) Computers, Electrocardiography and Public Health, USDHEW, P.H.S. Publication NO. 1644, 1967.

(7) Cathey, W.T., Bigelow, R.C., and Shander, D., Three Dimensional Display of a Vectorcardiogram, Proc. 7th Annual Rocky Mt. Bioeng. Sym., p. 30, 1970.

(8) Kubert,B., Szabo, J., Guilieri, S., The perspective representation of functions of two variables, J. of the Assoc. for Comp. Mach., Vol. 15, No. 2, April 1968, pp. 193-204.

(9) Standard Mathematical Tables,Chemical Rubber Co., Cleveland, Ohio, 1968.

ORIGINAL PAGE IS
OF POOR QUALITY